VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vytvoření nástrojů pro podporu práce s elektronickými zprávami

Development of Tools to Support Work with Electronic Messages

2018 Daniel Sedláček

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Daniel Sedláček**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vytvoření nástrojů pro podporu práce s elektronickými zprávami
Development of Tools to Support Work with Electronic Messages

Jazyk vypracování: čeština

Zásady pro vypracování:

Úkolem studenta je vytvoření internetového portálu, který bude sloužit k tvorbě a sdílení šablon elektronických zpráv - emailů. Při tvorbě portálu je nutné splnit následující úkoly:
1. Uživatel se může na portálu zaregistrovat a založit v něm svůj pracovní prostor. Do svého pracovního prostoru může pozvat další registrované uživatele, se kterými bude sdílet vytvořené šablony.
2. Uživatel může publikovat svou šablonu globálně, tedy i neregistrovaným návštěvníkům stránky. Ti mohou procházet jednotlivé kategorie šablon a zdarma si je stáhnout.
3. Portál umožní vytvoření šablony za pomocí WYSIWYG editoru. Šablona se automaticky ukládá do pracovního prostoru uživatele, který ji může kdykoliv editovat. Uživatel může v šabloně definovat místa pro dynamický obsah (content place holder).

Student také vytvoří strukturu knihoven tříd, které budou realizovat načtení emailové šablony a zaslání emailu. Knihovny tříd musí nabízet podporu následujících úkolů:
1. Podpora načtení předem vytvořené emailové šablony. Tato šablona bude načtena a všechna místa pro dynamický obsah budou nahrazena dodanými hodnotami.
2. Podpora zasílání mailů formou kopie, skryté kopie nebo hromadné zaslání mailu.
3. Uživatel bude schopen měnit nastavení poštovního serveru.
4. Student bude demonstrovat funkčnost práce na jednoduchém projektu, pro který postupně vytvoří šablonu emailu, kterou načte, naplní hodnotami a zašle na definované adresy.

Seznam doporučené odborné literatury:

[1] Troelsen, Andrew. Pro C# and the .NET 4.5 Framework 6th edition. Washington : Apress, 2012. 1430242337.
[2] Nitschke, Benjamin. XNA Game Programming. Indianapolis : Wiley Publishing Inc., 2007. 9780470126776.
[3] Bishop, Judith. C# 3.0 Design Patterns. Indianapolis : O'Reilly Media, 2007. 978-0-596-52773-0.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Plucar**

Datum zadání: 01.09.2014

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018

.....

Rád bych poděkoval Ing. Janu Plucarovi, Ph.D. za odborné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Abstrakt

Tato bakalářská práce se zabývá rozesíláním elektronických zpráv (emails) za pomocí šablon. Cílem je vyvinout nástroje, které tuto práci ulehčují a přidávají nové funkce oproti běžným nástrojům. Hlavní vlastností aplikace bude možnost tvorby šablon a její schopnost následného doplnění jedinečného obsahu jednotlivým příjemcům, který bude spjatý s tímto příjemcem. A to při odeslání jednomu či více příjemcům.

Práce se skládá ze tří hlavních aplikačních výstupů. Internetového portálu, na kterém uživatel provádí tvorbu a editaci šablon. Druhá část celého systému bude sada knihoven umožňující vytvoření a rozesílání zpráv. Ta na základě šablony vygeneruje nové elektronické zprávy. Třetí část bude desktopová a mobilní aplikace, která bude vycházet z desktopové verze.

Klíčová slova: elektronická zpráva, jedinečný obsah, šablona, internetový portál

Abstract

This bachelor thesis deals with sending emails using templates. The aim of this work is to develop tools that make this work easier and add new features to common tools. The main feature of the application will be the ability to create templates and its ability to subsequently add unique content to individual recipients that will be associated with that recipient. This is when sending to one or more recipients.

The work consists of three main application outputs. The Internet portal where the user performs the creation and editing of templates. The second part of the system will be a set of libraries that allow you to create and distribute messages. Based on the template, it generates an electronic messages. The third part will be a desktop and mobile application that will be based on a desktop version.

Key Words: email, unique content, template, internet portal

Obsah

Seznam použitých zkratek a symbolů

AES	–	Standard pokročilého šifrování (Advanced Encryption Standard)
CLS	–	Společná specifikace jazyka (Common Language Specification)
CSDL	–	Jazyk definice konceptuálního schématu (Conceptual schema definition language)
DCOM	–	Distribuovaný COM (Distributed Component Object Model)
DDL	–	Jazyk definice dat (Data definition language)
DLL	–	Dynamicky linkovaná knihovna (Dynamic-link library)
EDA	–	Architektura řízená událostmi (Event-driven architecture)
EDM	–	Entitní datový model (Entity Data Model)
FURPS	–	Funkčnost, vhodnost k použití, spolehlivost, výkon, schopnost být udržován (Functionality, Usability, Reliability, Performance, Supportability)
GUI	–	Grafické uživatelské rozhraní (Graphical User Interface)
HTML	–	Značkovací jazyk určený pro tvorbu webových stránek (HyperText Markup Language)
IIS	–	Internetové informační služby (Internet Information Services)
IS	–	Informační systém (Information system)
MSL	–	Jazyk specifikace mapování (Mapping specification language)
MSSQL	–	Microsoft SQL (Microsoft Structured Query Language)
MVC	–	Model, pohled, kontroler (Model View Controller)
ORM	–	Objektově relační mapování (Object Relational Mapping)
RAD	–	Rychlý vývoj aplikací (Rapid application development)
SDL	–	Jazyk definice schéma (Schema Definition Language)
SŘBD	–	Systém řízení báze dat
SSDL	–	Jazyk definice schéma skladu (Store schema definition language)
TCP	–	Protokol transportní vrstvy (Transmission Control Protocol)
UWP	–	Univerzální platforma Windows (Universal Windows Platform)
WYSIWYG	–	To, co vidíš, je to, co dostaneš (What you see is what you get) Jde o nástroj ke grafické editaci textového dokumentu v podobě, jak je dokument zobrazen, tak bude uložen.
XAML	–	Rozšiřitelný značkovací jazyk aplikací (Extensible Application Markup Language)
XML	–	Rozšiřitelný značkovací jazyk (Extensible Markup Language)

Seznam obrázků

Seznam tabulek

1 Úvod

Cílem práce je vytvoření aplikačních nástrojů umožňujících rozesílání elektronických zpráv, které vzniknou z předem vytvořených šablon, ty budou obsahovat uživatelsky definovaný dynamický prostor, tzv. Content place holder. Informační systém bude ulehčovat zasílání hromadných zpráv a automaticky generovaných zpráv nebo rozesílání zpráv, jejichž obsah se opakuje. Celá aplikace bude rozdělena do tří hlavních částí: internetového portálu a sady knihoven umožňující rozesílání elektronických zpráv, z kterých vznikne desktopová aplikace a mobilní klienti.

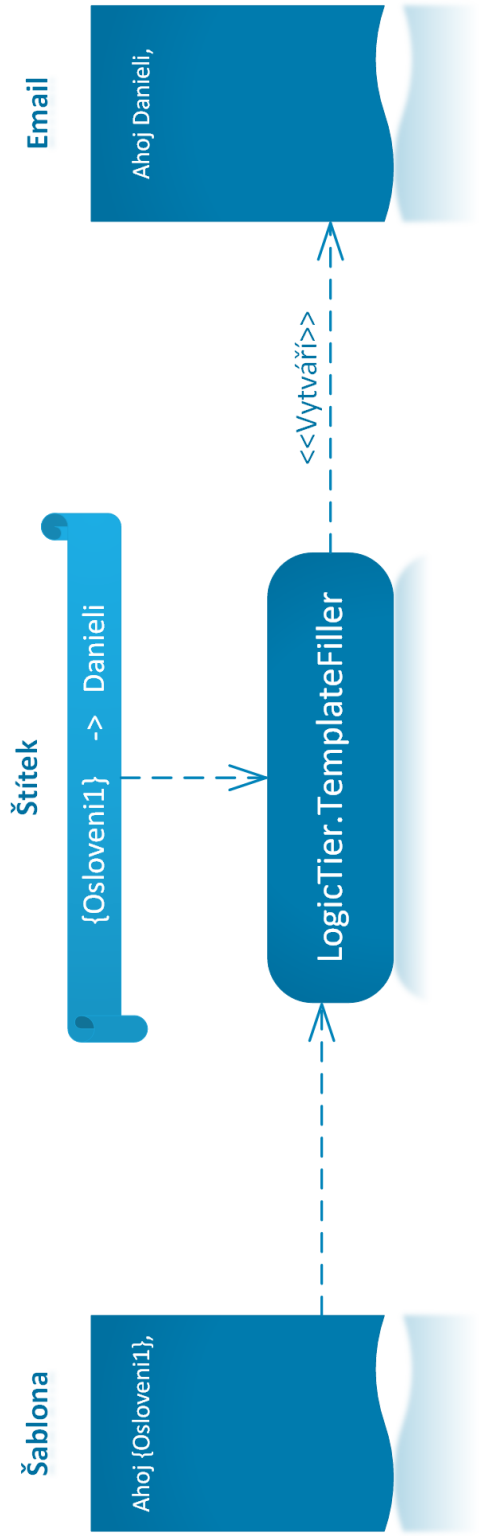
Na webovém portále se uživatel může zaregistrovat a spravovat svůj účet. Vytvářet a editovat šablony ve WYSIWYG editoru. Editor umožňuje editovat šablonu ve webovém prohlížeči ve vizuálním prostředí, za pomoci prvků editoru uživatel intuitivně přidává nový obsah, přesunutím prvku z menu na konkrétní místo v šabloně, např. tabulku, číslovaný nebo nečíslovaný seznam. Nebo mění vlastnosti stávajícího obsahu, např. změnu barvy prvku či jeho pozadí.

Uživatel může v šablonách definovat dynamický obsah (tzv. content place holder), který se před odesláním vyplní každému příjemci jedinečně na základě výběru štítku.

Štítek je množina dvojic, která naplní místa v šabloně, označená jako dynamický obsah, obsahem specifickým pro příjemce. Ve dvojici je vždy určeno místo k nahrazení a hodnota, která bude doplněna na toto místo. Schéma spojení šablony a štítku je zobrazeno na obrázku 1. Každé šabloně může být přiřazeno mnoho štítků. Uživatel tím získá možnost rozesílání hromadné informace zároveň s konkrétními údaji pro jednotlivé příjemce, např. s jeho jménem, názvem oddělení nebo s jménem nadřízeného či s číslem místnosti, ve které je referentka. Štítky se budou předem vytvářet a editovat na webovém portále.

Šablony jsou dvojího typu. Šablona je buď globální, to znamená veřejně přístupná pro všechny uživatele. Nebo privátní, tedy soukromá pouze pro registrovaného uživatele. Neregistrovaný uživatel si bude moct prohlížet a stahovat pouze globální (veřejné) šablony, tedy takové, které jiní registrovaní uživatelé označí jako globální a může tedy používat pouze již existující šablony. Registrovaný uživatel si může vytvářet a editovat šablony za pomocí integrovaného editoru. Dále získá vlastní pracovní prostor, do kterého může vytvořené šablony ukládat. Do tohoto prostoru může přizvat ostatní registrované uživatele, se kterými pak sdílí své privátní (soukromé) šablony. Uživatel může označit vybrané šablony jako globální, tak ať je vidí všichni uživatelé.

Všichni uživatelé si budou moct stáhnout sadu aplikačních knihoven umožňující vytvoření elektronické zprávy a její následné rozeslání. Tak bude aplikace integrovatelná do mnoha různých projektů, které jakkoliv používají odesílání elektronických zpráv. V této aplikaci se bude provádět samotná personifikace jednotlivých emailů, jejich specifické vyplnění na základě jednotlivého příjemce a šablony. Aplikace šablonu vyplní na základě předem vytvořeného štítku či množiny štítků, tím vznikne elektronická zpráva nebo více zpráv. Ty je pak možno pomocí integrovaného poštovního klienta rozeslat jednomu či více příjemcům. Dále si mohou uživatelé stáhnout desktopovou aplikaci nebo mobilní aplikaci, které budou tuto sadu knihoven využívat.



Obrázek 1: Schéma tvorby emailu z šablony

2 Metodika vývoje systému

Proces vývoje systému by se měl řídit nějakou množinou pravidel, doporučených postupů i různých nástrojů (frameworků), kterou obecně nazýváme metodika. Tato metodika se pak používá po dobu celého procesu, jak na začátku k vytvoření návrhu, dále pro organizaci plánování, tak pro samotné řízení procesu vývoje informačního systému. Metodika v sobě zahrnuje i specifické postupy pro pracovní tým nebo celou organizaci, dále obsahuje předběžnou definici výstupů a artefaktů, které budou součástí vytvořeného systému či následné údržby. Reálný proces vývoje nakonec využívá různé standardy, předepsané formáty, osvědčené postupy např. pro tvorbu kódu. Správnou volbou metodiky se sníží množství chyb a např. pravděpodobnost opomenutí části systému, která by pak znamenala velkou ztrátu času a prostředků. Nicméně metodika je pouze šablona, musí se přizpůsobit firmě, vývojovému týmu, projektu nebo bakalářské práci.

2.1 Rapid application development

Po posouzení zadání, rozsahu a typu této bakalářské práce byla pro organizaci práce, návrh a vývoj systému vybrána metoda **Rapid application development (RAD)** od Jamese Martina publikovaná v jeho knížce Rapid application development [3].

Obecně platí, že přístupy RAD k vývoji softwaru dávají menší důraz na plánování a větší důraz na proces. Koncepce RAD zdůrazňují přizpůsobivost a nutnost přizpůsobení požadavků v reakci na znalosti získané v průběhu projektu.

Jde o iterativní typ frameworku, který včasně reaguje na problémy při vývoji a zapracovává je do další iterace. V průběhu vývoje vznikají tzv. iterační verze, které nemají všechnu finální funkcionalitu, ale jsou spustitelné a testovatelné. Obvykle v krátkých vývojových cyklech, mezi jedním dnem až třemi týdny. Cílem těchto tzv. světlých verzí je důkaz existující koncepce pro klienta a hlavně konfrontace s uživateli pro zohlednění a doplnění jejich požadavků. Uživatelům je naopak umožněno poskytnutí zpětné vazby a mohou sledovat, jak vzniká čím dál tím více funkčnější aplikace a průběžně upřesňovat svoje požadavky. Klient, tedy zadavatel práce, je v tomto případě reprezentován vedoucím bakalářské práce. Obdobně vedoucí práce reprezentoval klíčové uživatele, iterativně zkoušející systém.

Je zaveden proces odkládání funkcí do budoucích verzí aplikací za účelem dokončení aktuální verze v co nejkratším čase. Striktní časový plán je důležitou součástí frameworku proto, aby byla zaručena kontinuální zpětná vazba, jejichž absence by celý vývoj prodloužila a zvětšila celkové náklady a čas.

Je doporučeno sestavování malých týmů, které jsou složeny z motivovaných a zkušených členů, kteří mohou zastávat více rolí. Vývojáři by měli znát nástroje a postupy použité v softwarovém inženýrství. V případě této práce byl jediný člen týmu autor práce a zastával tak všechny role.

Je nezbytně zavedeno aktivní řízení práce pro zmenšení rizik spojených s prodloužením vývojového cyklu. Dále pro prevenci vzniku nedorozumění mezi uživateli a vývojovým týmem.

Vedení týmu by se mělo zaměřit na výběr týmu a jeho motivaci, na odstranění byrokratických a politických překážek. Vedení týmu reprezentoval vedoucí práce.

Metoda RAD obsahuje čtyři rozlišné fáze vývoje.

2.1.1 Fáze plánování požadavků

První část spojuje prvky plánování systému a fáze analýzy. Obecné porozumění problémům, které obklopují vývoj, fungování a provoz systému. Identifikace obchodních procesů, které budou podporovány navrhovanou aplikací.

Na začátku zde probíhá řada setkání mezi klíčovými uživateli a vývojáři systému. Tato setkání slouží pro přípravu definice požadavků. Vývojáři systému se seznamují s aktuálním stavem tak, že porovnávají současné podobné systémy za účelem identifikace již existujících použitelných struktur.

Dále se v rámci této fáze vyvíjí model systému a navrhované systémové části. Funkčnost systému je vyjádřena v obchodních procesech a v datech, které bude systém podporovat. V průběhu vzniká obrys systémové oblasti a definice rozsahu systému, nakonec jsou však formálně zdokumentovány a schváleny zadavatelem. Vzniká odhad nákladů a doby implementace systému.

2.1.2 Fáze uživatelský design

Je vypracován podrobný model systémové oblasti. Jsou popsány všechny funkce systému k identifikaci opakovatelných prvků. Jsou specifikovány kritické funkce a jejich pořadí tvorby pro zachování řádné návaznosti při vývoji jednotlivých modulů systému. Vzniká předběžné uspořádání kritických zpráv a varování, které bude systém podporovat.

Návrhy, které vzniknou, jsou iteračně prozkoumávány členy vývojového týmu. Analýzou interakcí mezi funkcemi a daty se zjišťuje, jestli nechybí funkce pro existující data. Připraví se implementační plán systému. Součet odhadů úsilí potřebného k dokončení jednotlivých systémových částí ústí v celkový rozpočet. Na základě plánu jsou účastníci seznámeni s původně nepředvídanými problémy, neplatnými předpoklady v plánu a kritickými místy.

2.1.3 Fáze rychlá výstavba

Je navržena databáze na základě předběžné struktury dat shrnuté ve fázi uživatelský design. Je navržena strategie testování systému. Je nutné získání veškerého hardwaru nezbytného pro vývoj a provoz celé aplikace. Vznikne podrobná definice návrhu každé funkce na základě požadavků koncových a klíčových uživatelů.

Dále je nutno vyvinout aplikace potřebné k převodu dat z existujících formátů do formátů používaných aplikací. Začít iterativně zpracovávat dokumentaci celého systému. Připravit plán výcviku uživatelů a školící materiály pro to určené. Vypořádat se s organizačními a technickými problémy souvisejícími s nasazením nové aplikace. Důležitá je závěrečná kontrola konstrukce

systému, je provedena série testů, aby bylo ověřeno, že každá část systému a celý systém pracuje podle požadavků uživatelů.

2.1.4 Fáze přechod

V poslední fázi, pokud je potřeba, je zahájeno školení uživatelů na základě dokumentace získané ve fázi rychlá výstavba. Informace potřebné pro provoz nového systému se převedou z existujících zdrojů dat na formát dostupný pro nový systém. Dojde k nasazení systému. Dokončí se potřebné úpravy hardwaru a systémových aplikací, stejně jako prostředí interpretru, viz. kapitola 4.1 a všech používaných frameworků. Tento postup je zdokumentován, jehož důležitou součástí jsou verze jednotlivých komponent a je předán provoznímu týmu, který bude systém udržovat.

Nová instalace systému je přijata, pokud funguje po určitou dobu v rámci definovaných tolerancí výkonu, chybovosti a použitelnosti, viz. kapitoly 3.6.2, 3.6.3 a 3.6.4. Součástí tohoto přijetí je pak vzájemná dohoda mezi uživateli, zadavatelem, výrobním a provozním personálem.

Nasazení proběhlo v rámci této fáze pouze do testovacího prostředí. To znamená, že systém nebyl nasazen na reálný hostingový server.

3 Plánování požadavků

V této části se spojují prvky plánování systému s fází analýzy. Výrazně čerpá a pracuje se zadáním práce, jež je vlastně součástí této kapitoly v rámci metodiky vývoje.

3.1 Srovnání s existujícími produkty

Podobné produkty zabývající se hromadným rozesíláním emailů v dnešní době samozřejmě existují, každý se již s hromadně odeslaným emailem nejspíš setkal. V dnešní době se hromadně odeslané emaily nejvíce používají k rozesílání reklamních sdělení, většinou bohužel nevyžádaných doručiteli, tedy tzv. spamy.

Existující produkty stažitelné z Internetu byly ve volně šiřitelné verzi velmi jednoduché a nepodporovaly dynamické vyplnění obsahu příjemcům. Produkty, které toto umí, jsou placené a v bezplatné verzi jsou různě omezeny počtem odeslaných emailů, počtem příjemců. Na Internetu je také nabízena služba spojená s hromadným rozesíláním emailů s tzv. antispamovým filtrem, tedy úpravou textu a obsahu tak, aby u příjemců nebyla zpráva vyfiltrována jako nevyžádaná. Touto problematikou se tato práce nezabývá a není její součástí žádná úprava obsahu zprávy či nápověda jak obsah upravit.

Z konkurujících produktů byly vybrány nejrozšířenější aplikace pro porovnání a průzkum trhu. Aplikace SendBlaster 4, je distribuována skrze tlustého klienta a umožňuje hromadné rozesílání emailů, za pomoci šablon. Tlustý klient omezuje mobilitu, protože musí být předem před použitím na zařízení nainstalován, na to nemusí mít příslušný uživatel práva. V bezplatné verzi umožňuje vytvořit pouze 100 šablon, ve verzi placené jen 230. Nicméně v šablonách lze definovat pouze 15 předem definovaných dynamických prostorů, a to navázaných pouze na elektronickou nebo fyzickou adresu příjemce. Program umožňuje zaslat pouze 100 zpráv zdarma, poté umožní další odeslání až po koupi jediné placené verze Pro, ke které se ještě mohou dokoupit šablony. Cena trvalé licence placené verze Pro bez omezení je v rozmezí od 2574 Kč do 9074 Kč.

Designově dobře řešená webová aplikace `app.cakemail.com` umožňuje tvorbu šablon, jak ve vizuálním prostředí aplikace, tak jeho přímou editaci ve HTML kódu. Web umožňuje bezplatné stažení šablon. Nicméně v šablonách není možné definovat dynamický obsah. Tato aplikace je v neplacené verzi omezena možným počtem příjemců uložených v jednom účtu na 500. Pokud chcete tento počet rozšířit, musíte koupit vyšší verzi účtu, ta stojí od 184 Kč do 8878 Kč měsíčně.

Systém na rozesílání emailů na základě šablon sídlící na adrese `freemailtemplates.com` je zcela zdarma. Umožňuje tvorbu a editaci šablon, v nichž může tvořit dynamický obsah pouze email příjemce nebo informace, které nejsou přímo spjaté s příjemcem, např. aktuální čas a datum atd. Mezi srovnávanými produkty nabízí nejmenší funkčnost.

Při zhodnocení podobných produktů vyplývá, že systém, který by umožňoval neomezený počet uložených šablon s neomezeným počtem dynamických prostorů v šabloně, odesílaných neomezenému počtu příjemců s možností stažení šablon ostatních uživatelů zdarma a s možností rozšíření do dalších aplikací a vlastní desktopovou a mobilní aplikací, na trhu neexistuje.

3.2 Šablona elektronické zprávy

Šablona v systému slouží jako vzor pro vzniklé elektronické zprávy. Každá elektronická zpráva odeslaná pomocí systému vznikne z nějaké šablony systému. Šablonu může vytvářet pouze registrovaný uživatel a tedy ji i editovat a mazat. Každá šablona tak má svého autora, uživatele systému a pouze ten ji může editovat. Šablony rozlišujeme na globální a privátní. Privátní tedy soukromá je šablona implicitně, je tak viditelná pouze pro autora šablony. Ten ji může změnit na globální neboli veřejnou, ta je pak viditelná všem uživatelům systému a to i neregistrovaným návštěvníkům portálu, těm je tak umožněno vyzkoušení systému bez jakékoliv registrace. Ke správě a tvorbě šablon slouží internetový portál. Editace probíhá ve WYSIWYG editoru, který má standardní nástroje k editaci textu, změna formátu textu, editace fontu textu a barvy. Editor ukládá šablonu ve formátu HTML. Nicméně ukládaná data nesplňují povinnou strukturu jazyka HTML. Je zde vynechána deklarace typu dokumentu, kořenový element, hlavička dokumentu a uložená data obsahují pouze tělo dokumentu a to bez použití příslušné značky těla dokumentu. Šablona je uložena v HTML pouze pro potřeby formátování textu a vkládání dalších HTML prvků. Pokud by šablona obsahovala pouze čistý text, nemusí být její součástí HTML. Tato vlastnost může být využita např. při odesílání zpráv uživatelům, jejichž poštovní klienti nepodporují zobrazení ve HTML formátu.

V šabloně uživatel může vytvořit **uživatelsky definovaný dynamický prostor, tzv. Content place holder** a to v jakémkoliv množství. Ten slouží k personalizaci elektronických zpráv vzniklých z šablon. Jeho obsah bude při odesílání elektronických zpráv nahrazen obsahem příslušného štítku. Systém tak při každém odeslání zprávy vygeneruje zprávu novou z předem definované šablony a štítku. Při odesílání lze zvolit více štítků, tím systém vygeneruje na základě jedné šablony tolik zpráv jako štítků, množství takových štítků není nijak omezeno. Tyto zprávy jsou odeslány různým příjemcům.

Uživatelsky definovaný dynamický prostor se v šabloně vyznačuje množinovými závorkami, např. {osoba}, značí dynamický prostor s názvem osoba. Nicméně použití množivých závorek v běžném textu není nijak omezeno, pokud je uživatel použije a nechce je mít za dynamický prostor, stačí jejich obsah nedefinovat ve štítku a systém je bude ignorovat.

3.3 Dynamicky definovaný uživatelský prostor a štítek

Štítek slouží k nahrazení obecného dynamického prostoru, definovaného uživatelem v šabloně, hodnotami pro konkrétního uživatele pro konkrétní email. Je to množina takových dvojic, kde první z relace je dynamický prostor a druhá je hodnota, která ho nahrazuje. Ve štítku může být takových dvojic více, neomezeně mnoho. Ve štítku můžou a nemusí být všechny hodnoty dynamických prostorů obsažených v šabloně. Může jich být více či méně. Pokud systém nenalezne ve štítku hodnotu odpovídající dynamickému uživatelskému prostoru, ponechá dynamický prostor beze změny. Pokud ve štítku budou definovány hodnoty pro dynamické prostory, které nejsou v šabloně, systém je bude ignorovat. Jeden štítek tak může být použit k různým šablonám.

Štítek vzniká a edituje se na internetovém portále. Při odesílání emailu systémem uživatel určí šablonu a příslušný štítek, respektive jednu šablonu a množinu štítků při odesílání hromadného emailu. Vznikne tak množina jedinečných emailů, stejné mohutnosti jako původní množina štítků. Ty jsou následně odeslány různým uživatelům.

3.4 Pracovní prostor

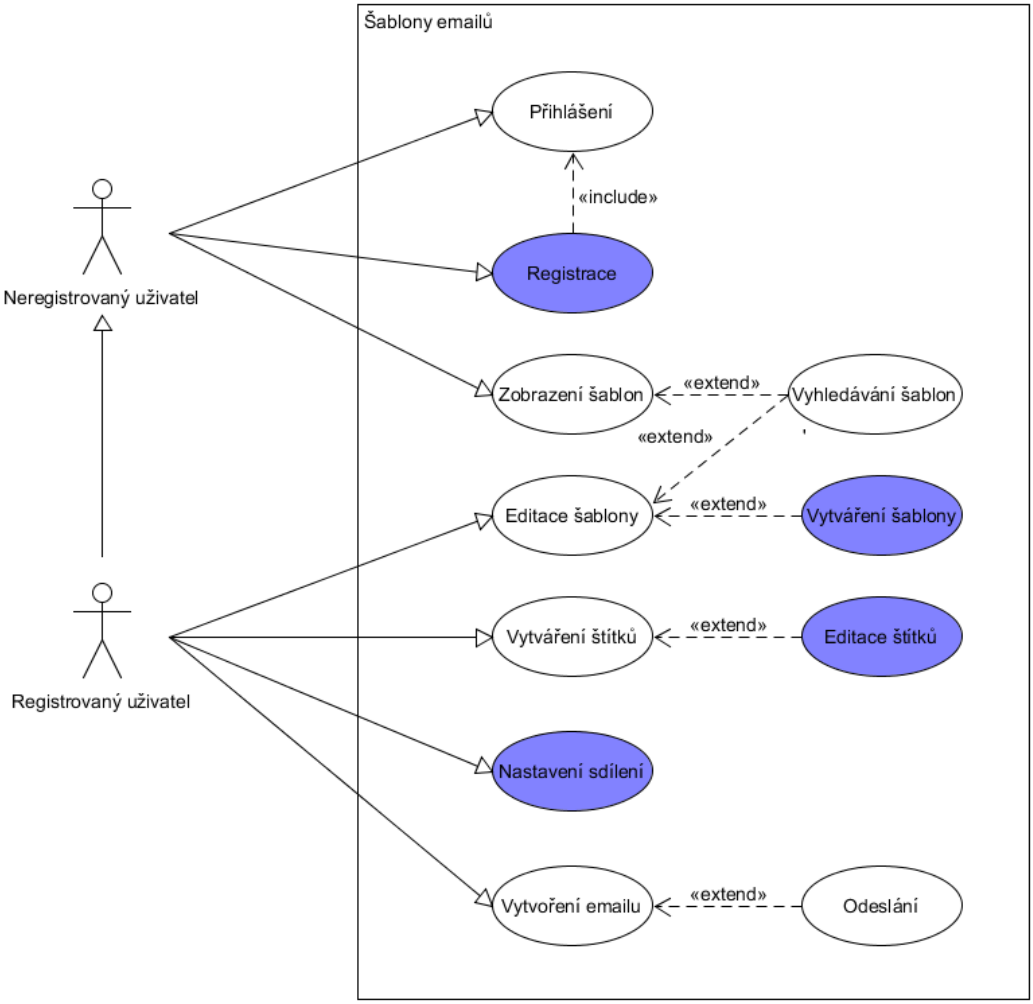
Na internetovém portále dále probíhá správa pracovního prostoru. Každý registrovaný uživatel může pozvat do svého pracovního prostoru jiné registrované uživatele. Počet takto pozvaných uživatelů není omezen. V rámci pracovního prostoru se sdílejí i privátní šablony. Pozvaný uživatel tak vidí všechny šablony uživatele. Sdílení je jednostranné, pokud jeden sdílející uživatel sdílí s druhým pozvaným, neznamená to, že uživatel pozvaný, také sdílí se sdílejícím. Pouze pokud by uživatel pozvaný, také pozval sdílejícího do svého pracovního prostoru.

3.5 Chování systému z pohledu uživatele

Chování systému z pohledu uživatele zachycuje diagram případů použití, tzv. use case diagram, viz. obrázek 2. Ukazuje nám, jaké funkcionality systém obsahuje, jak a kým jsou spouštěny. Skládá se z těchto následujících části:

Součástí každého takového diagramu je účastník, který je na diagramu reprezentován jako např. „Neregistrovaný uživatel“. Účastník reprezentuje kohokoliv či cokoliv mimo systém, tedy někoho, kdo se systémem komunikuje a interaguje. Jediné, co je účastníkovi dovoleno, je přijímat nebo předávat informace do systému. Účastník reprezentuje určitou roli, kterou může uživatel nabývat. Tedy jeden konkrétní uživatel interagující se systémem může být vyjádřen více účastníky, rolemi. A také více uživatelů může být vyjádřeno jedním účastníkem, rolí.

Další součástí diagramu je ohraničení, na diagramu zobrazeno obdélníkem s popiskem. Určuje hranice systému, obsah uvnitř obdélníku říká, jakou část či množinu funkcionalit informačního systému popisujeme. Uvnitř tohoto ohraničení se nachází samotná typová činnost (Use Case), na diagramu je zobrazena elipsou, ve které je popisek. Typová činnost určuje, jaké činnosti může účastník se systémem vykonávat. Pod diagramem následuje bližší specifikace jednotlivých činností, tzv. scénáře případu použití.



Obrázek 2: Diagram případů použití

3.5.1 Analýza klíčových procesů

Vybrané případy použití, podbarvené na diagramu, budou popsány přesněji v jejich scénářích. Popis bude zahrnovat jakou mají funkčnost a přidanou hodnotu pro uživatele, ukáže se důvod spuštění jednotlivých komponent. Dále jsou součástí scénáře aktéři, tedy objekty interagující s činností, většinou jde o uživatele nebo systém. Každý případ užití by měl mít definované podmínky spuštění, bez splnění těchto náležitostí činnost nemá smysl a nebude spuštěna. Hlavní scénář pak popisuje průběh scénáře. Popis by měl však být úplně odstíněn od toho, jak systém vypadá, měl by se zaměřit na to, jak funguje. Základní tok neřeší možné chyby a předpokládá bezproblémový průběh, kde v posledním kroku dojde ke splnění cíle případu užití. Naopak alternativní scénář umožňuje reagovat na odklonění od hlavního scénáře. Řeší případně vzniklé chyby nebo poruchy, jak na straně uživatele tak systému.

Tabulka 1: Scénář případu použití Registrace

Scénář případu použití: Registrace
Popis: Nový uživatel se zaregistruje v systému
Hlavní účastník: Neregistrovaný uživatel
Podmínky spuštění: Uživatel nesmí být přihlášen
Hlavní scénář: 1. Uživatel vyplní povinné popřípadě volitelné údaje k registraci 2. Uživatel potvrdí registraci a systém vytvoří registraci 3. Systém uživatele přihlásí
Alternativní scénář: Pokud nesouhlasí heslo nebo uživatel již existuje, údaje se musí upravit

Tabulka 2: Scénář případu použití Vytváření šablony

Scénář případu použití: Vytváření šablony
Popis: Uživatel vytvoří novou šablonu
Hlavní účastník: Registrovaný uživatel
Podmínky spuštění: Uživatel musí být přihlášen
Hlavní scénář: 1. Uživatel vybere vytvoření nové šablony 2. Mění její obsah pomocí WYSIWYG editoru, zadá název šablony a její viditelnost 3. Uživatel uloží šablonu

Tabulka 3: Scénář případu použití Editace štítků

Scénář případu použití: Editace štítků
Popis: Uživatel změní obsah štítku
Hlavní účastník: Registrovaný uživatel
Podmínky spuštění: Uživatel musí být přihlášen a mít již vytvořenou šablonu a k ní štítek
Hlavní scénář: 1. Uživatel vybere štítek k editaci 2. Přidává nové položky štítku, nový klíč a jeho hodnotu 3. Uživatel uloží štítek s novými položky
Alternativní scénář: Uživatel neuloží změny

Tabulka 4: Scénář případu použití Nastavení sdílení

Scénář případu použití: Nastavení sdílení
Popis: Uživatel změní přístup ke svým privátním šablonám
Hlavní účastník: Registrovaný uživatel
Podmínky spuštění: Uživatel musí být přihlášen
Hlavní scénář: 1. Zobrazí seznam všech uživatelů systému a vidí, jestli s daným uživatelem již sdílí šablony 2. Uživatel přidává a odebírá sdílení s ostatními uživateli

3.6 Analýza požadavků

Požadavky jsou zaváděny proto, aby bylo specifikováno to, co by mělo být následně naimplementováno. Požadavky se rozlišují dvojího typu, funkční a nefunkční požadavky. Funkční požadavky určují jaké chování bude systém mít. Nefunkční požadavky specifikují vlastnosti nebo omezující podmínky pro daný systém. Jedna z možností, jak pohlížet na systém, jsou FURPS požadavky. Metodika FURPS rozděluje požadavky do pěti různých kategorií, které jsou: funkční požadavky, požadavky na vhodnost k použití, spolehlivost, výkon, schopnost být udržován.

3.6.1 Funkční požadavky

Funkční požadavky se zaměřují na hlavní funkčnost a schopnosti systému.

Název Odeslání emailu
Popis: Uživatel může pomocí knihovny tříd odeslat email jednomu nebo více příjemcům se zadanou šablonou a štítkem k jejímu vyplnění.
Priorita: 10

Název Editace šablony
Popis: Uživatel bude moci na webovém portále prohlížet a editovat všechny své šablony ve WYSIWYG editoru.
Priorita: 10

Název Nastavení účtu
Popis: Uživatel bude moci na webovém portále měnit svoje heslo, jméno a příjmení.
Priorita: 7

Název Vytvoření štítku
Popis: Uživatel bude moci na webovém portále ke konkrétním šablonám vytvářet štítky .
Priorita: 10

3.6.2 Vhodnost k použití

Vhodnost k použití se hodnotí zejména z pohledu uživatele. Kritéria jsou celková použitelnost systému, uživatelská ovladatelnost.

Název Personalizace prostředí
Popis: Uživatel bude moci na webovém portále personalizovat prostředí, měnit barvu jednotlivých ovládacích prvků a jejich polohu.

Priorita: 5

Název Uživatelsky přátelské prostředí
Popis: Uživatel bude intuitivně ovládat celý IS, lehce se zorientuje ve všech ovládacích prvcích.
Priorita: 8

Název Platformově nezávislé prostředí
Popis: Celá webová část systému bude plně responzivní pro možnost použití na různých koncových zařízeních.
Priorita: 9

3.6.3 Spolehlivost

Jedná se o hodnocení četnosti a závažnosti chyb, přesnosti zpracování vstupů a výstupů. V této oblasti se také sledují možnosti obnovení provozu a zotavení se z výpadku. Patří sem také zátěžové testy a testy zotavení aplikace po selhání některých komponent řešení.

Název Odolnost vůči chybám a neočekávaným ukončením aplikace
Popis: Systém bude odolný proti špatně zadaným vstupům a proti neočekávaným výjimkám, např. nefunkčnost databáze.
Priorita: 9

Název Nepřetržitý provoz
Popis: Systém bude možné provozovat nepřetržitě.
Priorita: 10

Název Obnovení systému
Popis: Po neočekávaném ukončení aplikace nebo databáze bude systém možné zprovoznit do 8 hodin.
Priorita: 8

3.6.4 Výkon

Hodnotí se rychlost systému, celková odezva a vytíženost hardwarových prvků. Podle toho se určí vhodný hardware pro daný systém.
Název Rychlé načítání webových stránek systému
Popis: Všechny webové stránky by se měly načítat do 2 sekund.

Priorita: 5

Název Obsluha více uživatelů
Popis: Systém by měl být schopen obsluhovat až 500 uživatelů současně.
Priorita: 7

3.6.5 Schopnost udržování

Posledním hlediskem hodnocení je oblast údržby systému. Hodnotí se možnosti testování aplikace, možnosti podpory. Další možnosti rozšiřitelnosti systému o nové prvky.

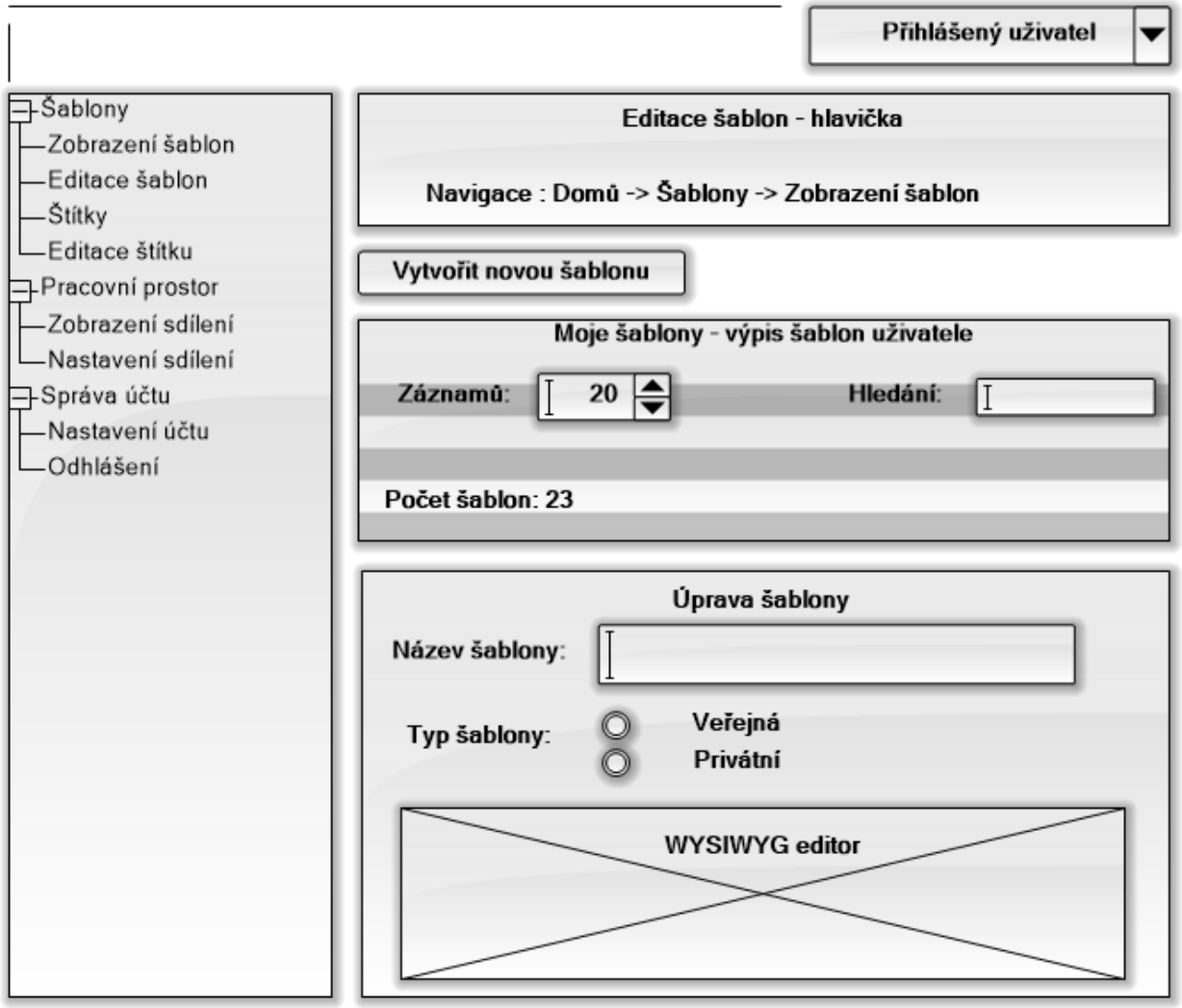
Název Provoz webové části v síti Internet
Popis: IS musí být nasaditelný v reálném provozu na Internetu.
Priorita: 10

Název Rozšiřitelnost systému
Popis: IS musí být lehce rozšiřitelný o další funkcionalitu aniž by stávající funkčnost byla výrazně měněna.
Priorita: 10

3.7 Grafický návrh internetového portálu

Návrh je zobrazen pomocí skici, která definuje rozmístění funkčních prvků na stránce a používá se pro náhled nového systému. Ukazuje návrh funkcí a obsah stránek webového systému pro lepší pochopení mezi uživateli a vývojáři. Slouží k minimalizaci rozdílu mezi originálním zadáním a dodaným řešením. Určuje, kde bude menu pro výběr akcí, kde bude hlavní okno aplikace.

Rozvržení webu je demonstrováno na obrázku 3. Vlevo se nachází menu, ve kterém se uživatel přepíná mezi částmi systému. Toto menu se dá zmenšit a místo popisků s ikony jsou vidět pouze ikony. Vpravo nahoře se nachází rozbalovací uživatelské menu, kde uživatel najde uživatelské funkce jako odhlášení a správa účtu. V centrální části obrazovky najdeme hlavní část aplikace, a to na základě toho, v jakém menu se nacházíme. Zde jsou seznamy výpisů šablon, atd. a také možnost samotné editace ve WYSIWYG editoru. Na skice lze vidět rozvržení obrazovky pro menu k editaci šablon. Obsahuje seznam již vytvořených šablon s možností vyhledávání, s tlačítkem pro tvorbu nové šablony a hlavně okno, určené pro úpravu samotné šablony s WYSIWYG editorem.



Obrázek 3: Skica webové části systému, přihlášený uživatel a editace šablony

4 Uživatelský design

V této kapitole se navrhne softwarová architektura systému, popíše se podrobně její struktura, komponenty a způsob jejich návaznosti. Popíšeme vývojové prostředí a použité frameworky. V této fázi se iteračním vývojem dosáhlo finální podoby, specifikované v předchozí kapitole. Je popsána závěrečná iterace, či spíš všechny proběhlé iterace v cyklu.

4.1 Popis implementačního prostředí

Pro vývoj byl vybrán programovací jazyk **C#** navržený pro vytváření různorodých aplikací, které běží v rozhraní .NET Framework. Jazyk C# je výkonný, typově bezpečný a objektově orientovaný. Řada inovací v jazyce C# umožňuje rychlý vývoj aplikací a zároveň zachování expresivity a elegance jazyků stylu C. Proto jsem si tuto technologii vybral pro tvorbu této bakalářské práce.

Visual C# je implementace jazyka C# společností Microsoft. Visual Studio podporuje kód Visual C# a je plně vybaveno editorem, kompilátorem, šablonami projektu, výkonnými a snadno použitelnými ladícími programy a dalšími nástroji. Knihovna tříd rozhraní .NET Framework poskytuje přístup k řadě služeb operačního systému a dalším užitečným a dobře navrženým třídám, které umožňují výrazné urychlení vývoje.

Historie C# je neoddělitelně spjata s platformou Microsoft .NET. V roce 1993, kdy byly uvedeny modely COM (Component Object Model), softwarová architektura umožňující vystavět aplikace z binárních komponent a DCOM (Distributed Component Object Model), protokol umožňující komunikovat softwarovým komponentám přes síť včetně Internetu pomocí HTTP protokolu. Tyto dva modely se postupem času v roce 1997 spojily do COM+, což bylo rozšíření původního COM rozhraní. To umožňovalo softwarovým komponentám psaným v různých jazycích, aby spolu komunikovaly. Tato myšlenka byla použita roku 1999, kdy se Microsoft zmiňuje o technologii .NET, platformě skládající ze z tzv. CLR (Common Language Runtime) a z knihovny tříd systému .NET Framework někdy nazývaných Base Class Library (BCL). CLR má být společným běhovým prostředím pro programy psané v různých jazycích, díky CLR spolu mohou jednotlivé části programu (nebo celé programy) navzájem spolupracovat, přestože byly napsány každý v jiném programovacím jazyce. Toho je dosaženo tím, že aby programovací jazyk mohl produkovat kód kompatibilní s CLR, musí vyhovovat specifikacím CLS (Common Language Specifications). A zde právě přichází na řadu C#. Je to první jazyk navržený přesně podle systému CLS a tím umožňuje asi nejlépe využít veškeré jeho možnosti. Přestože s jistými úpravami vyhovuje specifikaci CLS více programovacích jazyků, které jsou pro prostředí .NET upravovány, C# vznikl přesně pro toto prostředí. Pokud kompilujete program v C#, výsledkem není přímo strojový kód proveditelný procesorem, ale kód v jazyku MSIL (Microsoft Intermediate Language), který je teprve pomocí CLR přeložen do nativního kódu procesoru. Nejedná se tedy o jazyk interpretovaný jako třeba Java, ze které se C# hodně inspiroval, ale ve výsledku přímo o spustitelný kód, i když generovaný automaticky až těsně před spuštěním programu.

Přestože první beta verze byla hotová již v roce 1999, 26. června 2000 ohlašuje Microsoft uvedení jazyka C# a počátek jeho standardizace. V první polovině roku 2001 je dokončen kompilátor pro jazyk C# a celá platforma .NET 1.0 je oficiálně uvolněna pro veřejnost. Momentálně je aktuální verze .NET Framework 4.7.1.

Z popisu jazyka C# tak vyplývají následující výhody. Je založen na historickém úspěchu jazyků C/C++. Tím je ověřeno, že syntaxe jazyka umožňuje vytvářet pro vývojáře přehledné konstrukce různorodých programů, které jsou i po mnoho let udržovány a aktualizovány. Nicméně oproti C/C++ přináší řadu vylepšení. Jde o platformově nezávislý jazyk. Spravuje automatické přidělování a uvolňování paměti aplikace. Jedná se o masově a široce používaný jazyk s garantovanou podporou.

4.2 Softwarová architektura systému

Architektur pro návrh systému je mnoho a liší se použitím, velikostí systému a strukturou. Vybral jsem několik nejvhodnějších možných architektur pro tento systém a porovnal je.

Architektura řízená událostmi (Event-driven architecture) je model založený na vzniku události a následné reakci na tuto událost. Událost lze definovat jako změnu stavu, např. přihlášení uživatele, vytvoření nové šablony. Taková událost může být sdělena i jiným aplikacím, které jsou součásti architektury. Informace, která se produkuje, publikuje, propaguje nebo rozpoznává, se nazývá zpráva, jde tedy o oznámení o události, ne událost samotnou. Zprávy vznikají zpravidla asynchronně. Komponenty v systému buď vysílají zprávy nebo na ně reagují. Zprávy se obvykle zasílají přes síť prostřednictvím HTTP nebo prostřednictvím jiných protokolů. Architektura řízená událostmi má jednu nebo více front, do kterých jsou všechny zprávy vloženy před zpracováním. Zpráva ve frontě zůstává nastálo nebo může být odebrána. Sběratelé odebírají zprávy z konkrétních front a reagují tak na konkrétní události. Budování systému architekturou řízenou událostmi umožňuje menší odezvu, protože konstrukce systému je založena na reakci nepředvídané události.

Prostorová architektura (Space-based architecture) je navržena pro širokou škálovatelnost a řešení souběžných událostí. Návrh odstraňuje z architektury úzká místa webových aplikací, typicky je takovým místem databáze. Architektura odstraňuje centrální databázi a nahrazuje ji sdílenou pamětí. Každý objekt obsahující informace systému je uložen v tzv. datové mřížce, v té je uložena mimo jiné časová známka objektu. Datové mřížky jsou součástí jednotky, tedy prvku aplikace, který vykonává určitou činnost. Jednotka může svoji datovou mřížku replikovat a poskytnout ji jiné jednotce. Tím je umožněno sdílení dat napříč systémem bez centrálního úložného bodu. Řeší problém většiny webových aplikací, které i přes zvyšování hardwarových prostředků systému, narazí na svoje maximum odbavených požadavků. Tato architektura je vhodná pro rozsáhlé a velké projekty s velkým počtem uživatelů, dotazů a prováděných operací.

Opačný přístup pro uložení dat než předešlá architektura využívá **databázově orientovaná architektura (Database-centric Architecture)**, ta pro ukládání informací používá standardních systémů pro správu relačních databází. Využívá elementů a entitních typů data-

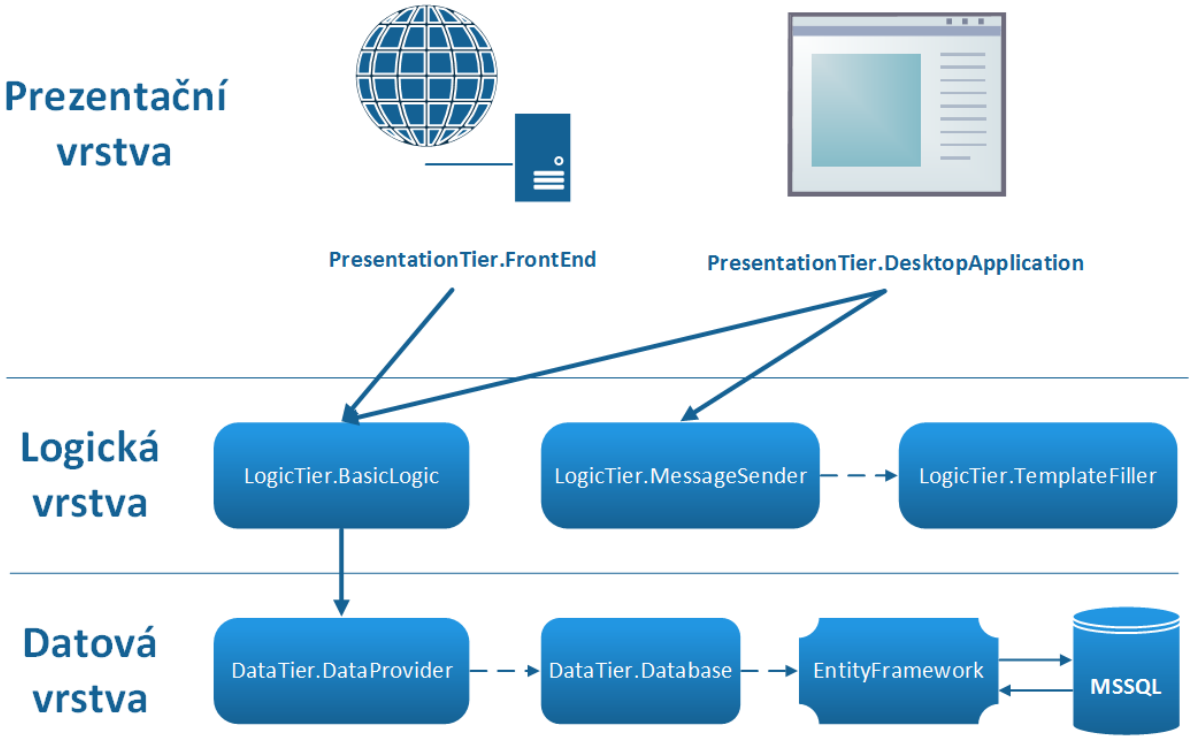
báze a omezuje použití struktur doménové logiky aplikace. Pracuje s relační logikou řízenou tabulkami, tedy chování a procesy jsou silně ovlivněny obsahem a strukturou databáze. Práce s daty je založena na funkcích a procedurách uložených na relačních serverech oproti aplikační logice systému. Využívá sdílenou databázi jako základ pro komunikaci mezi paralelními procesy v distribuovaných informačních systémech, na rozdíl od předávání zpráv. Zjednodušuje návrh systému pomocí zpracování veškeré logiky aplikace na databázovém serveru a zmenšuje tak velikost a složitost aplikací uživatelských klientů tenkých i tlustých. Relačně databázový server je kritickým místem architektury, kdy jakákoliv chyba v návrhu struktury databáze nebo při provozu serveru bude nákladně odstraňována.

Vícevrstvá architektura, také známá jako **n-vrstvá** , kde n značí počet vrstev, se používá v aplikacích, kde je zastoupen klient a server. Jednotlivé vrstvy se pak rozdistribuují mezi klientem a serverem. Hlavní motivace architektury je logické rozdělení programu do jednotlivých vrstev, které vždy odděleně plní svoji úlohu. Systém je díky toho přehlednější při vývoji. Jednotlivé funkce můžou být rozmístěny na fyzicky různých zařízeních, což umožňuje poskytování služeb bez nutnosti sdílení zdrojů, a jsou tak dostupné v dostatečné kapacitě. Vznikne tak ale množina různorodých hardwarových prostředků. A operační systémy nasazené na více strojích budou v různých verzích i architekturách. Což nutně znesnadní vývoj, nasazení a hlavně následnou údržbu, jak v časových tak finančních parametrech. To lze eliminovat nasazením virtualizační technologie, ale pouze částečně, webový server a antivir se musí aktualizovat neustále.

Při následném upravení nebo rozšíření, které jsou v rámci životního cyklu programu nezbytné, lze vždy upravovat jen jednu část tak, aby ostatní zůstaly nezměněny, pokud zůstane původní rozhraní. Vytváření vrstev by tak nemělo být závislé na ostatních vrstvách. Vrstva samotná komunikuje vždy jen s vrstvy přímo sousedícími. Je důležité, aby bylo přesně definováno jakým způsobem mezi sebou vrstvy komunikují. Nelze říct jaký počet vrstev je optimální. Vždy se to odvíjí od konkrétních potřeb uživatelů a celkové stavby systému.

4.2.1 Třívrstvá architektura

Pro tento systém byla vybrána jako nejvhodnější třívrstvá architektura. Třívrstvá architektura rozlišuje vrstvu datovou, logickou a prezentační. Na obrázku 4 je diagram, na kterém je znázorněno jednotlivé rozdělení aplikace do vrstev a propojení vrstev mezi sebou. Architektura se typicky používá u webových aplikací. Odděluje grafické prostředí v nejvyšší vrstvě, které může mít mnoho podob, klientů. Je zde webová část aplikace nebo klasická desktopová aplikace. Ve prostřední vrstvě je umístěna nejcitlivější logika aplikace tak, aby byla nezávislá na zbytku systému. V nejnižší vrstvě se pak pracuje s daty na nejnižší úrovni, bez znalosti jejich významu a vazeb. Následuje popis jednotlivých vrstev a jejich funkce v systému.



Obrázek 4: Diagram třívrstvé architektury

4.3 Datová vrstva

Také se nazývá databázová, nejnižší vrstva systému. Zajišťuje práci přímo s daty, používá základní datové funkce. Zajišťuje celkové uložení informací a další práci s nimi. Její účel je zajistit celkový přístup k datům tak, aby se ostatní vrstvy nemusely starat o to, jaká databáze je použita atd. Pokud se změní databáze nebo typ úložiště, změna proběhne pouze v datové vrstvě.

V bakalářské práci zajišťuje v IS komunikaci s MSSQL databází, jde o relační databázi od firmy Microsoft. K přístupu k datům jsem se rozhodl použít Entity Framework (EF). Pro implementaci byl vybrán návrhový vzor Data Mapper publikovaný Martinem Fowlerem ve své knize Patterns of Enterprise Application Architecture, viz. [4]. V informačním systému je datová vrstva reprezentována dvěma projekty: DataTier.DataProvider a DataTier.Database.

4.3.1 Entity Framework

Je sada technologií v ADO.NET, které podporují vývoj datově orientovaných softwarových aplikací a umožňuje vývojářům pracovat s relačními daty jako s objekty bez nutnosti zajímat se o databázové tabulky, kde jsou tato data uložena. Entity Framework umožňuje vývojářům pracovat s daty ve formě doménově specifických objektů, aniž by bylo nutné se zajímat o strukturu databáze, kde jsou tato data uložena. S Entity Frameworkem mohou vývojáři pracovat na vyšší úrovni abstrakce, když přistupují k datům a mohou vytvářet a udržovat aplikaci datově oriento-

vanou s menším množstvím kódu. Entity Framework je součástí .NET Framework od verze 3.5 SP1.

Entity Framework používá vlastní formát modelu reprezentace dat. Nazývá se Entitní datový model (Entity Data Model). V projektu pojmenovaný jako DatabaseModel.edmx, který obsahuje všechny artefakty a metadata. Popis entit a jejích vazeb je vyjádřen jazykem definice schéma (Schema Definition Language), který je typu XML. Je složen ze tří modelů.

Konceptuální datový model vyjádřen jazykem definice konceptuálního schématu (Conceptual schema definition language), který obsahuje jednotlivé entity. Ty jsou tvořeny agregací více polí. Toto pole je konkrétní sloupec z relační databáze. K tvoření entity můžou být použita pole z různých tabulek. Vazby, které jsou v modelu popsány, jsou také nezávislé na vazbách v databázi.

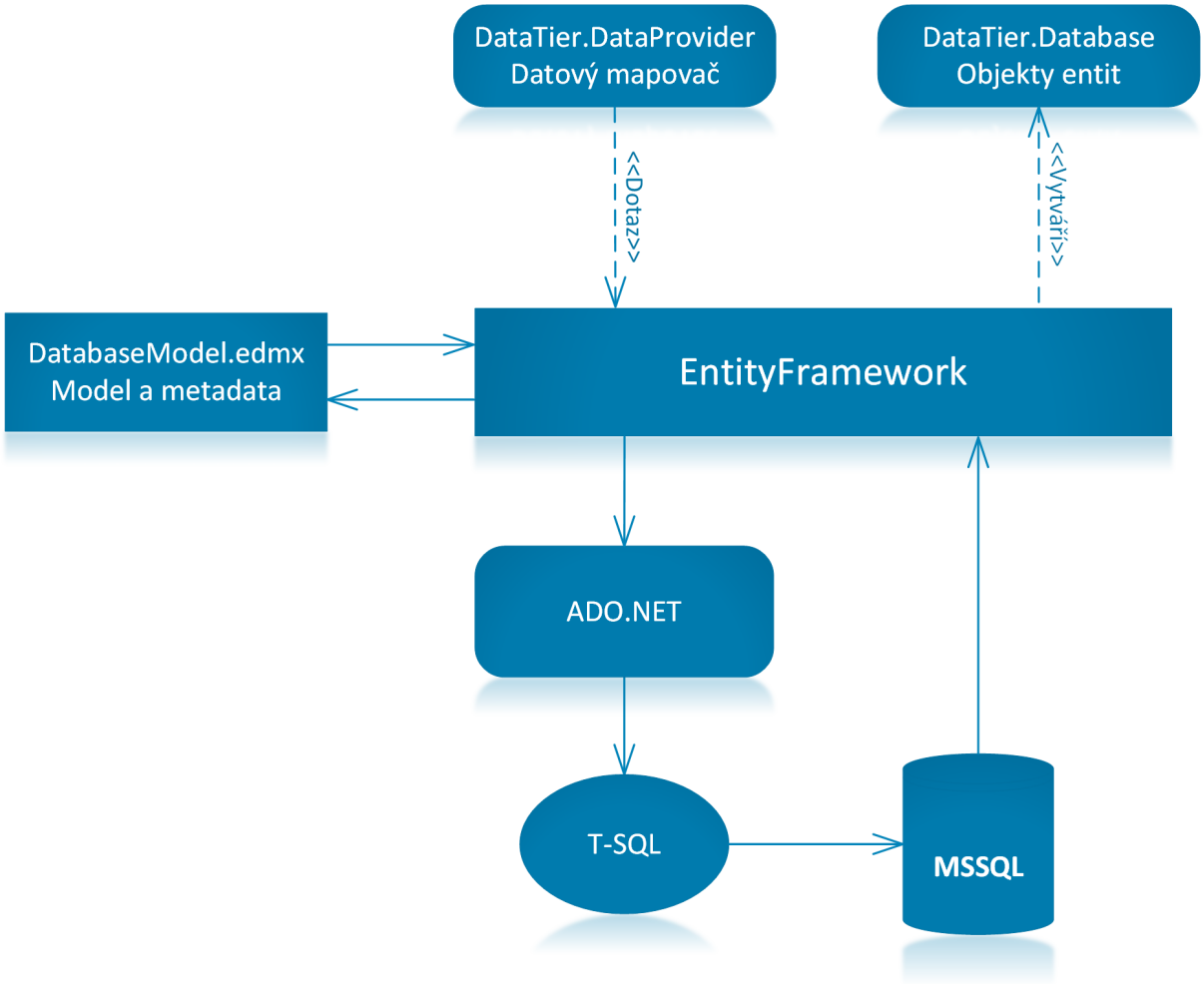
Takto popsané entity tvoří množinu dat strukturovaných v doménové podobě aplikace. Vznikne z nich tak množina tříd, generovaných Entity Frameworkem, určených pro přímé použití aplikací. Vztahy entit v těchto třídách jsou řešeny seznamy s názvy těchto vztahů. Každá entita bude mít automaticky vlastní index jako primární klíč. Instance těchto tříd budou obsahovat konkrétní data z databáze. Tyto instance budou obsahovat množinu objektů entitního typu modelu s indexovacím klíčem.

Dále skladovací model (Storage model), který je vyjádřen jazykem pro definici schéma skladu (Store schema definition language). Ten popisuje hierarchické uspořádání entit a jejich entitních typů a vazeb mezi nimi v relační databázi.

Poslední model je definovaný v jazyku specifikace mapování (Mapping specification language). V něm se předchozí dva modely spojují dohromady, obsahem modelu je mapováním konceptuálního schématu na skladovací model. Vznikne tak ucelený popis dat od relační úrovně až po doménové objekty.

ADO.NET Entity Framework definuje svoji vlastní variantu strukturovaného dotazovacího jazyka pojmenovaného Entity SQL. Ten je svoji strukturou určen k psaní deklarativních dotazů, tedy dotazů, ve kterých pouze deklarujeme, která data chceme získat a je na systému, aby je získal, tedy zjistil, jak je má získat. Vztahy mezi entitami jsou zapsány na konceptuální úrovni, tím se liší od SQL, protože používá konceptuální datový model. Takto vytvořený dotaz přijímá třída EntityClient, sestaví z něj SQL dotaz a vrácená data z databáze ukládá do instancí tříd vytvořených z konceptuálního modelu.

Na obrázku 5 je vidět schematické rozmístění komponent.



Obrázek 5: Diagram Entity Framework

4.4 Logická vrstva

Tato vrstva, také nazývaná jako aplikační vrstva, se nachází na aplikačním serveru. Tvoří spojení s ostatními vrstvami. Tato vrstva sousedí s prezentační a databázovou vrstvou. Měli by zde probíhat právě všechny operace, které přímo nesouvisí s načítáním a ukládáním dat. V logické vrstvě se spojují informace z datové vrstvy a vyhodnocují a formátují se pro prezentační vrstvu. Na této vrstvě se provádí kromě samotných výpočtů a vyhodnocování i autentizace, autorizace uživatele a i personalizace jeho profilu. Ve vrstvě je umístěn vzor abstraktní továrna pro tvorbu elektronických zpráv. V mém systému je logická vrstva tvořena čtyřmi projekty: LogicTier.BasicLogic, LogicTier.MobileLogic, LogicTier.EmailSender a LogicTier.TemplateFiller.

4.5 Prezentační vrstva

Je nejvyšší vrstva informačního systému. Interaguje přímo s uživatelem a měla by tomu být uzpůsobena, jde tedy o část systému, kde nás zajímá nejen funkčnost ale i uživatelská ovladatelnost. Vrstva zpracovává informace přímo od uživatele a pomocí logické vrstvy je realizuje, dále zobrazuje informace uživateli z logické vrstvy. Jde většinou o program nasazený na webovém serveru jako webová aplikace nebo o uživatelskou aplikaci ve stolním počítači nebo mobilním zařízení.

V prezentační vrstvě je webový projekt jako hlavní část, se kterou interaguje uživatel. Projekt bude nasaditelný na webovém serveru. Vrstva generuje HTML obsah pro prohlížeče uživatelů na základě zaslaných dotazů. Ve webové části prezentační vrstvy je použit vzor **MVC (Model View Controller)**. Dále je zde umístěna desktopová a mobilní aplikace poštovního klienta, která umožňuje vyplnění a odeslání šablony. Vyplněním šablony vznikne elektronická zpráva, k tvorbě takové zprávy je použita abstraktní továrna umístěna v logické vrstvě. V tomto systému je prezentační vrstva tvořena webovým projektem PresentationTier.FrontEnd, desktopovou aplikací PresentationTier.WindowsApplication a třemi mobilními klienty PresentationTier.Mobile.Android, PresentationTier.Mobile.iOS a PresentationTier.Mobile.UWP.

4.6 Prvky architektury

Jednou z možností tvorby objektově orientovaných aplikací je využití návrhových vzorů (Design Patterns). Návrhové vzory nepocházejí ze softwarového inženýrství, jsou zcela běžné v každodenním životě. Vzory obecně popisují, jak řešit abstraktní problém, a je úkolem vývojáře v modelu upravit prvky vzoru tak, aby odpovídaly požadavkům aktuálního problému.

Jedná se o popis nebo šablonu, jak řešit problém, který lze použít v mnoha různých situacích. Pro každý problém existuje mnoho vzorů, který jej řeší, je na návrhovém týmu vybrat ten nejvhodnější s ohledem na zapojení do systému. Společné vzory návrhu mohou být časem zlepšeny, což je robustnější nástroj než použití návrhů specializovaných k účelu.

Používají se pro zlepšení kvality systému, protože jsou vytvořeny na základě dobrých řešení častých problémů. Nemění však funkcionalitu systému, ale pouze organizaci nebo strukturu systému. Návrhové vzory mohou urychlit proces vývoje tím, že poskytnou testovaná osvědčená vývojová paradigmata. Návrhové vzory vytvářejí obousměrný most mezi teorií a praxí tím, že spojují praktickou zkušenost s teorií a přenášejí teorii do praxe. Jde o skupinu spolupracujících tříd, jejíž propojením popsaném ve vzoru dosáhneme typizovaného řešení problému.

Opětovné použití návrhových vzorů předchází vzniku malých problémů, které mohou způsobit velké změny softwaru a analýzy. Použití návrhových vzorů nám umožňuje napsat kód, ve kterém je pro jiné vývojáře a architekty snazší se orientovat. Při správném použití dělají kód udržovatelnější, snadnější pro čtení i vývoj. Zkušenosti z praxe ukazují, že znalost návrhových vzorů skutečně zjednodušuje práci a jejich adaptace pomáhá zlepšovat strukturu výsledných

zdrojových kódů. Někteří kritici však namítají, že se jedná o zbytečné konstrukce obcházející nedokonalosti některých imperativních programovacích jazyků.

4.7 Vzor datový mapovač

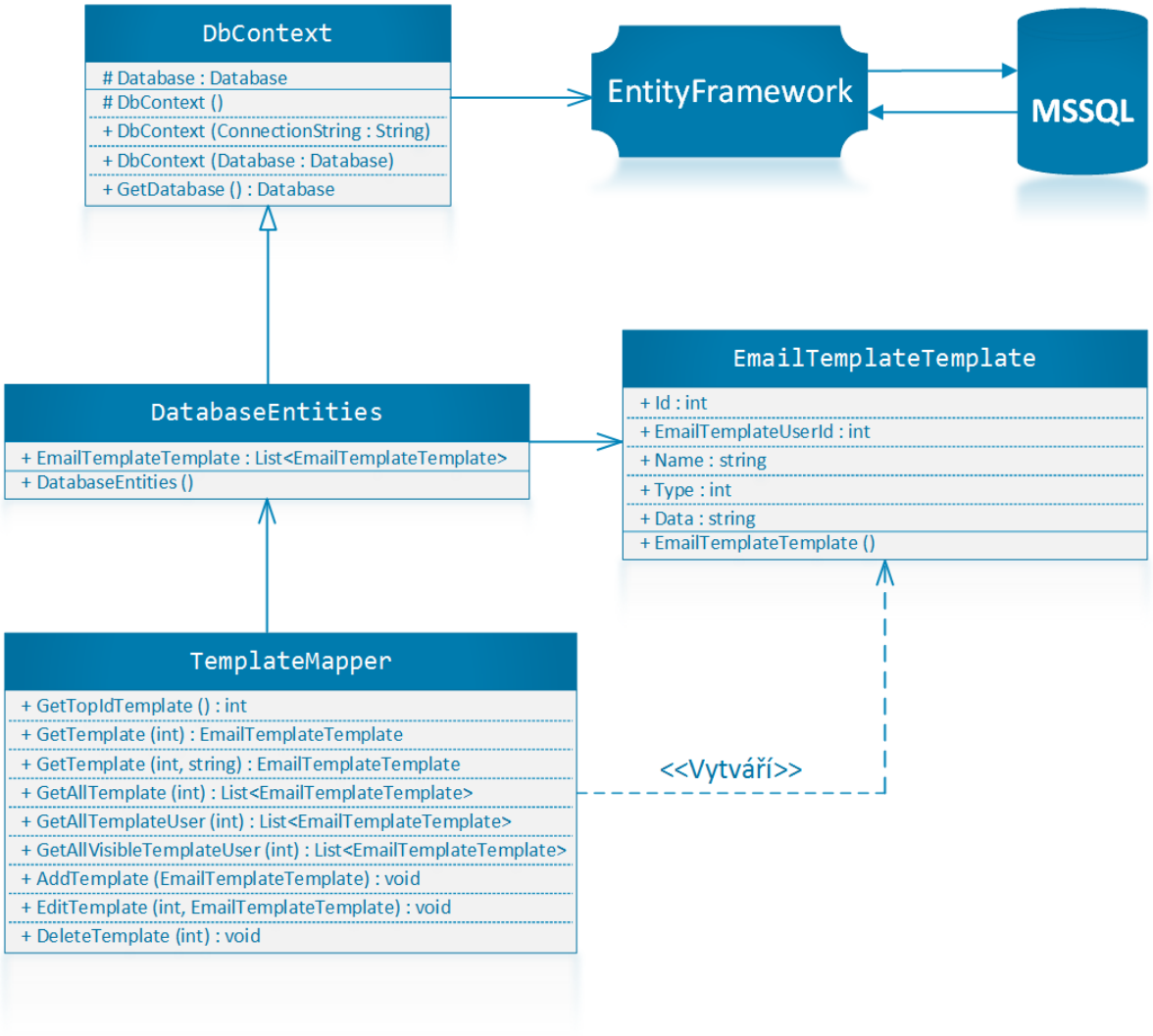
Třídy aplikace definované v programovacím jazyce a jejich následná struktura v paměti je odlišná od struktury uložení dat v relační databázi. Při tvorbě databáze a její struktury entit a formátu entitních typů vzniká naprosto odlišná reprezentace významově stejných dat než v aplikaci. Je třeba přenášet tato data mezi oběma systémy a to oboustranně.

Vzor datový mapovač (Data Mapper) tyto různorodé objekty dvou systémů spojuje, byl zveřejněn Martinem Fowlerem v jeho knize o návrhových vzorech [4]. Jeho úkolem je přenášet data mezi systémy a izolovat je od sebe. Aplikace nemusí znát strukturu ani způsob manipulace dat v databázi a naopak. Nachází se v datové vrstvě systému.

Rozhraní objektu, které odpovídá tomuto vzoru, by mělo zahrnovat metody, jak vyhledávat a číst, upravovat a odstraňovat objekty, které odpovídají doménovým entitám datového úložiště. Vznikne tak pohodlná krátkodobá forma dat pro manipulaci v aplikaci na jedné straně a na straně druhé možnost tato data trvale zálohovat v databázi. Vrstva obsahuje jeden nebo více mapovačů, každý je pak obvykle zodpovědný za manipulaci s jednou entitou. V tomto systému jde o projekt DataTier.DataProvider.

Mapovač za použití Entity Framework obsahuje konkrétní dotazy na databázi. Ty jsou zapsány ve strukturovaném dotazovacím jazyce Entity SQL, jazyk Entity Framework. Dotazy jsou důležitým prvkem mapovače, formulují jaká data chceme získat. Mapovač pak tyto dotazy předá Entity Framework a vrátí očekávaná data. Nakonec se musí tato data převést do podoby přijatelné aplikací. Vznikne instance objektů vhodných pro manipulaci v systému a naplní se hodnotami ze získaných dat. Struktura získaných dat nemusí být vždy stejná jako chtěná finální podoba. Proto plnění instancí není vždy triviální záležitost, může zde vzniknou indexovací tabulka pro rychlejší vyhledávání navrácené množiny objektů. Tyto naplněné instance objektů doménové logiky aplikace nakonec mapovač vrací volajícímu objektu.

Datový mapovač a jeho ukázka použití v IS je zobrazena na diagramu 6.



Obrázek 6: Diagram vzoru Data Mapper v systému

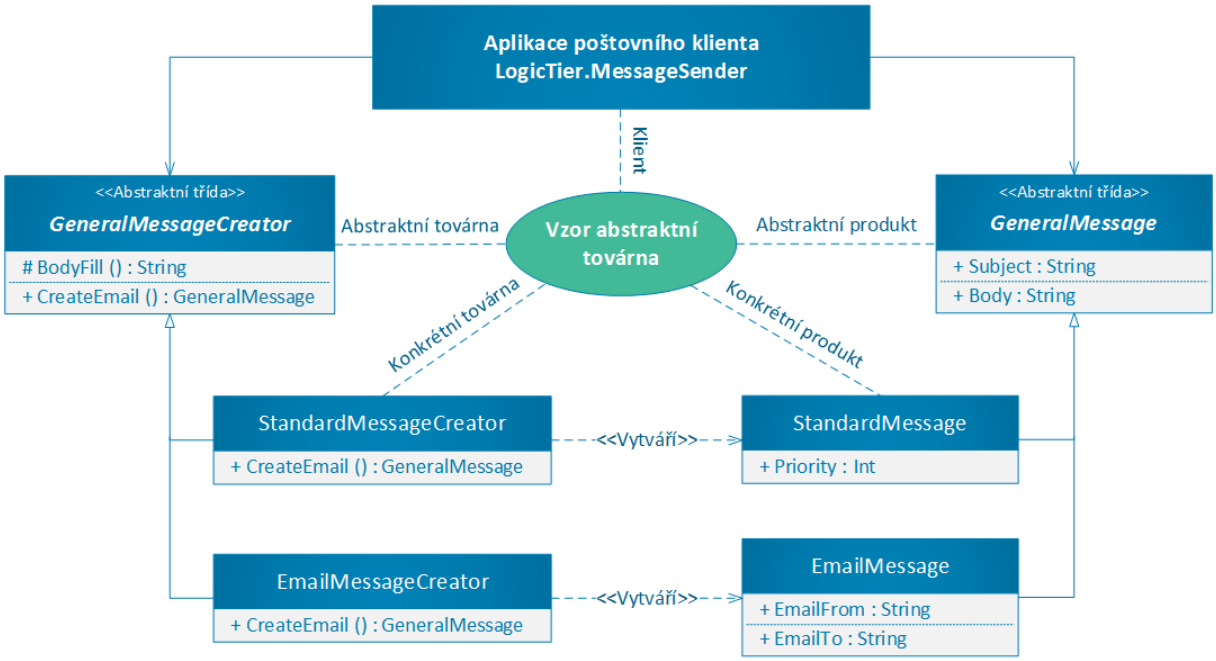
4.8 Vzor abstraktní továrna

Jeden z nejlepších způsobů, jak vytvořit objekt, je za využití návrhového **vzoru továrna**. Ten umožňuje tvorbu vyšší abstrakce při tvorbě objektu třídy než standardní konstruktor. Typicky se používá k zapouzdření složitější inicializace instance a pro vytváření různých typů instancí podle parametrů. Můžeme v továrně umístit proměnné, které slouží k vytváření objektu. Tyto proměnné pak nemusí být součástí zbytku systému, kde tyto objekty vznikají. Obvykle se používá návratová hodnota pro vrácení objektu, který nemusí být přesně typ objektu vytvářeného, ale může být typu některé rodičovské třídy či rozhraní. Cílem továrny je odstínit klienta od vytváření nových objektů.

Vzor abstraktní továrna ukazuje způsob sjednocení skupiny továren, které mají podobný účel. A to bez specifikace svých konkrétních tříd. Běžně se vzor používá tak, že klientský soft-

ware implementuje rozhraní abstraktní továrny v konkrétní verzi. Pak používá obecné rozhraní továrny k vytvoření a manipulaci s objektem. Klientské části aplikace se tedy nemusí zajímat o specifické objekty, které jsou navráceny, protože používá pouze obecné rozhraní. Ve vzoru jsou odděleny detaily v implementacích různých sad objektů a jejich obecné manipulace s nimi, při tom se spoléhá na složení objektu, protože jejich vytváření je implementováno metodami vystavěnými pouze na základě společného rozhraní.

V systému je použití vzoru využito v aplikaci poštovního klienta při tvorbě konkrétních emailů. V abstraktní třídě *GeneralMessageCreator* je deklarována podoba všech ostatních továren, které budou vytvářet konkrétní instance produktů, elektronických zpráv. A je zde definována jedna metoda, která je společná pro všechny implementující třídy továren, na tvorbu těla zprávy. Dále existuje abstraktní produkt, elektronická zpráva. Jde o abstraktní třídu *GeneralMessage* obsahující společné prvky všech zpráv, předmět a tělo zprávy. Tuto abstraktní třídu produktu implementují jednotlivé produkty, tedy jejich třídy, které reprezentují různé typy elektronických zpráv. Jedna implementace továrny vytváří instance jednoho typu produktu, továrna je tedy uzpůsobena na tvorbu určitého produktu elektronické zprávy. Objekt, který chce zprávu vytvořit, si nejdříve musí vytvořit příslušnou továrnu a ta mu pak vytvoří konkrétní elektronickou zprávu typu odpovídajícímu továrně. Na obr. č. 7 je zobrazen třídní diagram rozvržení vzoru v konkrétní podobě použité v systému.



Obrázek 7: Třídní diagram použití abstraktní továrny v aplikaci poštovního klienta

4.9 Vzor MVC

Model View Controller (MVC) je velmi oblíbený návrhový vzor, který se uchytil zejména na webu, ačkoli původně vznikl na desktopech. Je součástí populárních webových frameworků, jakými jsou např. Zend nebo Nette pro PHP, Ruby On Rail pro Ruby nebo MVC pro ASP.NET. Ten je také použit v tomto projektu.

Základní myšlenkou MVC architektury je oddělení logiky od výstupu. Řeší tedy problém tzv. špagetového kódu, kdy jsou v jedné třídě logické operace a zároveň renderování výstupu. Celá webová aplikace je rozdělena na komponenty 3 typů, hovoříme o modelech (Model), pohledech (View) a kontrolorech (Controller). U MVC neexistuje žádná striktní definice architektury, a tak se můžeme setkat s více výklady. Použil jsem ten nejrozšířenější.

Komponenta model z MVC architektury. Model obsahuje logiku a vše, co do ní spadá. Mohou to být výpočty, databázové dotazy, validace a podobně. Model vůbec neví o výstupu. Jeho funkce spočívá v přijetí parametrů zvenku a vydání dat ven. Model neví, odkud data v parametrech přišla a ani, jak budou výstupní data zformátována a vypsána. Instance modelů obsahují atributy z databáze nebo od uživatele k uložení. V tomto projektu je model umístěn v druhé logické vrstvě jako projekt LogicTier.BasicLogic.

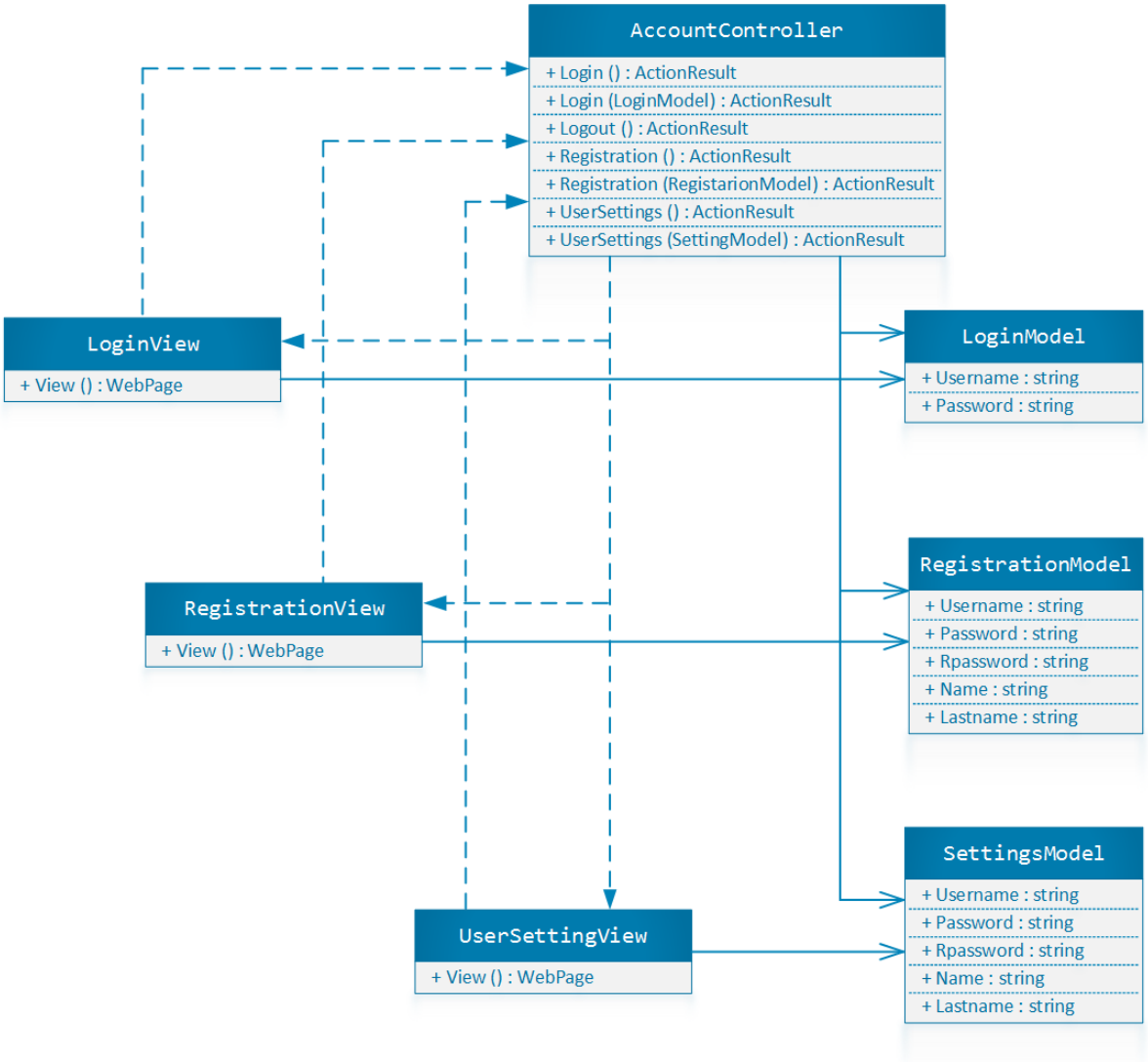
Komponenta view z MVC architektury. Pohled se stará o zobrazení výstupu uživateli. Nejčastěji se jedná o HTML šablonu obsahující HTML stránku a tagy značkovacího jazyka, který umožňuje do šablony vkládat proměnné, případně provádět iterace a podmínky. Šablony lze samozřejmě vkládat do sebe. Pohled není jen šablona, ale zobrazovač výstupu. Obsahuje tedy minimální množství logiky, která je pro výpis nutná, např. kontrola, zda si uživatel vyplnil přezdívku před jejím vypsáním. Pohled podobně jako model vůbec neví, odkud mu data přišla, stará se jen o jejich zobrazení uživateli.

Komponenta controller z MVC architektury. Kontrolér je nyní onen chybějící prvek, který osvětlí funkčnost celého vzoru. Jedná se o jakéhosi prostředníka, se kterým komunikuje uživatel, model i pohled. Drží tedy celý systém pohromadě a komponenty propojuje.

Požadavek uživatele jako první zachytí tzv. router. Ten podle parametrů pozná, který kontrolér je volán. Daný kontrolér podle parametrů pozná, co se po něm chce, např. že má zobrazit detail uživatele. Zavolá model, který uživatele vyhledá v databázi a vrátí jeho údaje. Dále zavolá další metodu modelu, která např. vypočítá věk uživatele. Tyto údaje si kontrolér ukládá do proměnných. Nakonec vyrenderuje pohled. Název pohledu kontrolér pozná podle akce, kterou provádíme. Pohledu jsou předány proměnné s příslušnými daty. Pohled přijme data od kontroléru a vloží je do připravené šablony webové stránky. Hotová stránka je zobrazena uživateli. Kontrolér a pohled jsou umístěny v prezentační vrstvě v projektu PresentationTier.FrontEnd.

MVC architektura usnadňuje i myšlení při vývoji projektu. Když se píše logika, umístí se do modelu. Formátování a stylování výstupu se řeší v pohledu. To, co uživatel chce z parametrů, se zjistí v kontroléru. Tři různé problémy na třech různých místech oddělené tak, aby do

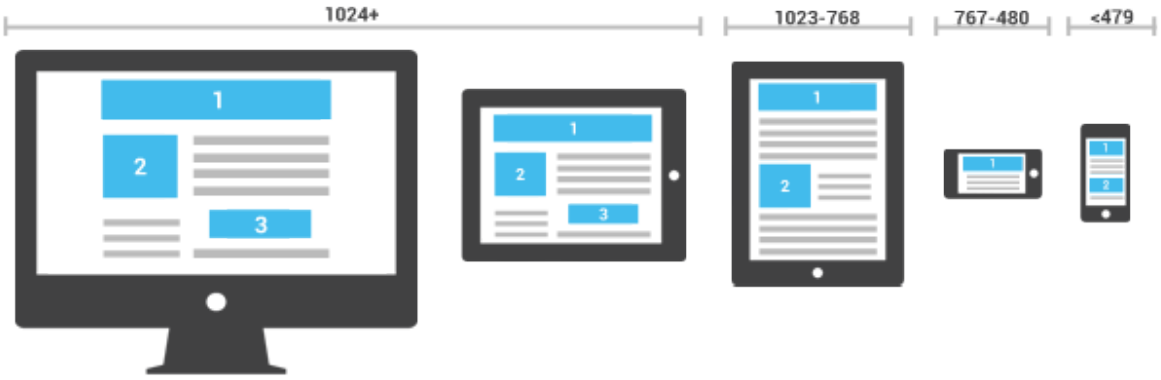
sebe nezasahovaly a nekomplikovaly tak vývoj systému. Na diagramu 8 je zobrazena návaznost jednotlivých komponent systému.



Obrázek 8: Diagram zapojení MVC v systému

4.10 Responzivní webový design

Responzivní webový design zabezpečuje optimální zobrazení webových stránek na různých typech zařízení jako je počítač, tablet, chytrý telefon. Rozložení prvků na stránce se mění v závislosti na velikosti použitého displeje. Výhodou je existence pouze jedné verze webových stránek. Administrátor nebo majitel webových stránek dělá všechny úpravy na jednom místě a bez ohledu na cílové zařízení. Responzivní design se postará o jejich správné zobrazení na každém zařízení. Z technického hlediska se jedná o přizpůsobení CSS šablon pomocí tzv. Media Querries. Jde o pravidla, kterými lze měnit rozvržení dokumentu v závislosti na aktuální šířce zobrazované plochy. Jednotlivé prvky na stránce se pak poskládají a přizpůsobí do layoutu různým způsobem podle typu zařízení, ze kterého uživatel na web přistupuje. Webové stránky se tak přizpůsobí velikosti displeje zobrazovacího zařízení, vše se děje automaticky, tedy bez nutnosti obnovit stránku.



Obrázek 9: Nákres responzivního zobrazení webové stránky

5 Rychlá výstavba

Pro vývoj celého systému jsem se rozhodl použít vývojové prostředí od společnosti Microsoft Visual Studio 2015 Enterprise, vývoj mobilní aplikace, finální nasazení a kompilace proběhla v aktuální verzi Visual Studio 2017 Enterprise. V něm jsem použil webový projekt obsahující technologii ASP.NET MVC 5. Prostředí podporuje práci s Entity Frameworkem, konkrétně vizuální a intuitivní prostředí pro tvorbu a editaci EDM modelů pro definici dat. Vývojové prostředí bylo vybráno, protože umožňuje spojit vývoj webové aplikace, desktopové aplikace poštovního klienta a mobilní klienty. Pro ukládání dat při vývoji byl rovněž vybrán produkt téže společnosti. Relačně databázový server Microsoft SQL Server 2016.

5.1 Webový projekt

Pro tvorbu internetového portálu byl zvolen framework ASP.NET MVC 5, jež je obsažen ve webovém projektu Visual Studia. Ve projektu je již vygenerována základní struktura adresářů webové aplikace, které odpovídají architektuře MVC. Framework v sobě obsahuje logiku vzoru MVC a jeho řídící prvky.

Kontroléři jsou umístěni ve složce Conrollers, jde o třídy, které jsou potomky třídy Controller. Jeden kontrolér reprezentuje jednu stránku internetové aplikace a je napsán v jedné třídě. V takové třídě jsou metody, jejichž návratovými typy je ActionResult, což je objekt, který reprezentuje odpověď, která je po dokončení požadavku zaslána zpět prohlížeči. Může jít o webovou stránku nebo o přesměrování na jinou adresu. Metody jsou všechny přetížené, mají stejný název, protože reprezentují jednu stránku, jejich vstupní parametry se liší na základě toho, jaké formuláře byly na internetové stránce vyplněny a odeslány aplikaci. Pokud přijde HTTP požadavek na internetovou aplikaci MVC framework pomocí routeru rozhodne, jaký kontrolér bude vybrán a jaká metoda v něm bude zavolána k obsloužení požadavku.

Ke každému kontroléru vznikne jeho vlastní složka pro jeho pohledy ve složce Views. Zde se umísťují šablony webových stránek. Jsou tedy tvořeny kombinací značkovacího jazyka HTML a zdrojového kódu C#. Pro předávání dat mezi kontroléry a pohledy slouží kolekce ViewBag. Je přístupná jak v kontroléru, kde se naplní daty k zobrazení, tak v pohledu, kde dochází k jejímu formátovanému zobrazení. Kolekce využívá vlastnosti dynamických objektů, které nám umožňují místo použití klíčů zapisovat přímo do vlastností.

Modely jsou umístěny ve složce FormModels. Jde o třídy, které reprezentují po datové stránce formulář webové stránky, takový formulář, který je definovaný v nějakém pohledu systému. Pokud je vyplněn formulář na webové stránce a zaslán v HTTP požadavku aplikaci, MVC framework naplní příslušnou třídu modelu. Tato třída poté funguje jako vstupní parametr odpovídající metody kontroléru. A tím se data z formuláře dostanou již typově a velikostně ošetřena ke zpracování a vyhodnocení.

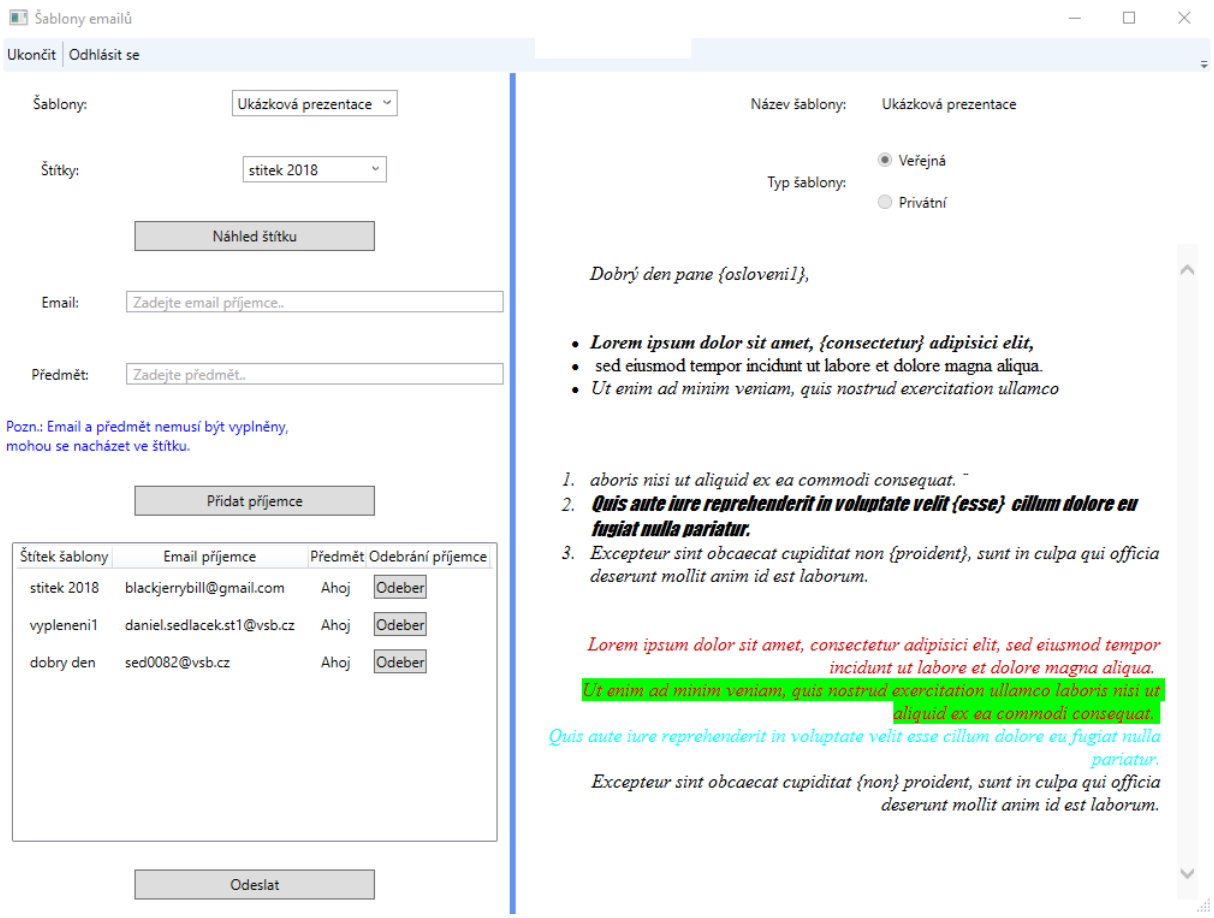
5.2 Desktopová aplikace

Další součástí systému je desktopová aplikace tlustého klienta, která vyžívá WPF (Windows Presentation Fundation). WPF je technologie pro vývoj grafického prostředí od společnosti Microsoft, které je součástí .NET Frameworku od verze 3.0. WPF využívá k vykreslování Direct3D, jež je rozhraní specifické pro systémy Microsoft Windows využívající hardwarovou akceleraci grafických adaptérů.

Rozložení grafických elementů je definováno relativně vzhledem k ostatním elementům. Vnitřní reprezentace grafických elementů je vektorová. Tyto vlastnosti usnadňují vývoj aplikací pro různá rozlišení pracovních ploch. K popisu grafického rozhraní, definici rozložení grafických elementů a k definici jejich vizuálních vlastností jako velikost, barva, viditelnost se ve WPF používá soubor typu XAML (Extensible Application Markup Language). XAML je značkovací jazyk, který odpovídá standardu XML. Takový popis se dá zapsat i ve formě zdrojového kódu a je nezbytné ho umístit do souboru se zdrojovým kódem při dynamických změnách grafických prvků v rámci běhu aplikace. Nicméně použití XAMLu umožňuje přehledný náhled na grafickou podobu aplikace, přehlednější rozdělení vývoje při více početných týmech. Za dobrou praxi se považuje umístění co nejvíce popisu do XAML souboru na úkor zdrojového kódu. Tento postup je použit i při vývoji této aplikace.

Pro předávání dat grafickým elementům se ve WPF používá Datová vazba (Data Binding), která poskytuje flexibilní, jednoduchý a konzistentní mechanismus pro synchronizaci dat v databázi či jiném úložišti a uživatelského rozhraní. Jde o vazbu, jejichž vykonání zajišťuje WPF framework, vazba může být jednosměrná v obou směrech nebo obousměrná. Grafické elementy v tomto projektu využívají k přístupu k datům datovou vazbu, ta je navázána na modely Entity Frameworku z datové vrstvy, které plní metody z logické vrstvy. Aplikace obsahuje všechny potřebné knihovny systému datové i logické vrstvy, jedná se o desktopovou aplikaci tlustého klienta, ta minimalizuje komunikaci po síti, která probíhá pouze mezi aplikací a databázovým serverem.

Po spuštění aplikace se zobrazuje přihlašovací obrazovka. Po úspěšném přihlášení aplikace načte všechny viditelné šablony uživatele a první z nich mu zobrazí. Uživatel může listovat mezi šablonami a vybírat, kterou bude odesílat, editace je možná pouze na webovém portále. Uživatel může přidávat adresy jednotlivých příjemců emailů a k nim vybírá jeden ze svých štítků, též zobrazených v aplikaci. Takových příjemců může být neomezeně mnoho a jeden příjemce může být přidán vícekrát s různými štítky. Aplikace umožňuje náhled jednotlivých štítků pro šablony a uživatel tak získá podobu konkrétního emailu doručenému uživateli. Při odesílání aplikace na základě šablony a množiny štítků vygeneruje stejně mohutnou množinu emailů a pokusí se je odeslat na příslušné adresy, počet úspěšně odeslaných emailů se zobrazí po odeslání.



Obrázek 10: Náhled na desktopovou aplikaci po přihlášení

5.3 Mobilní aplikace

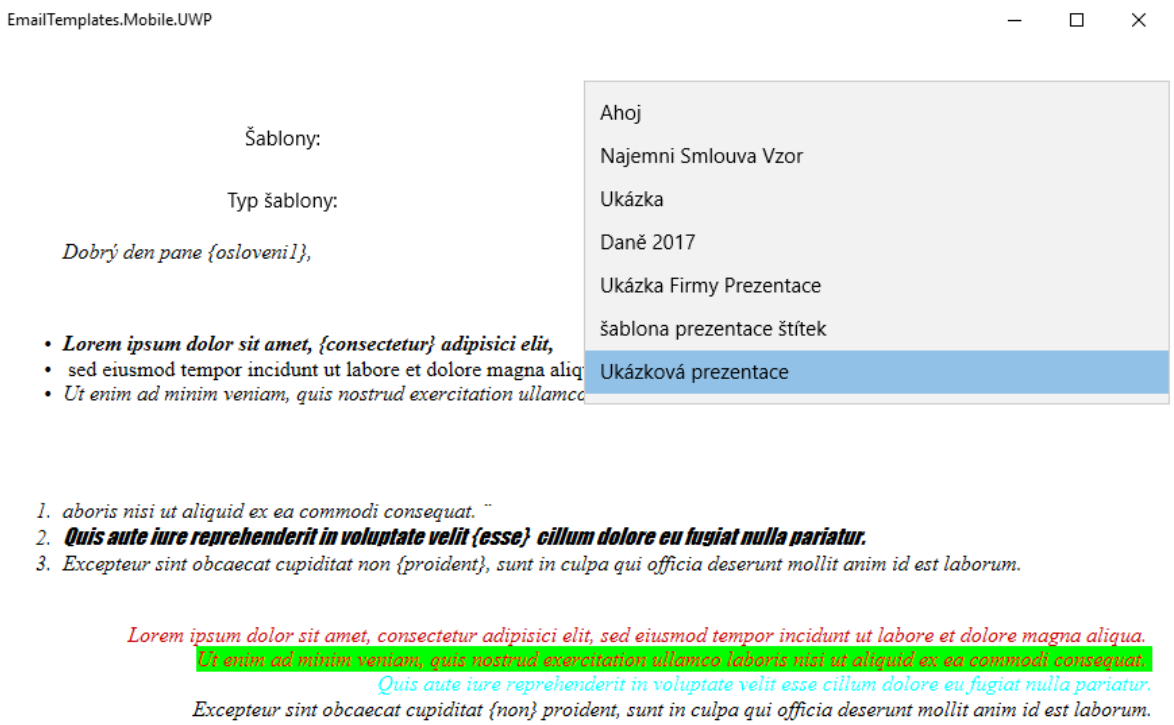
K tvorbě mobilního klienta byla vybrána technologie Xamarin.Forms, která je součástí Visual Studia 2017. Technologie umožňuje vyvíjet aplikace cílené na Android, iOS a UWP (Universal Windows Platform). Universal Windows Platform je univerzální platforma, která poskytuje společnou platformu pro aplikace na každém zařízení, na kterém je nainstalován systém Windows 10. Aplikace cílené na UWP můžou být spuštěné na různých zařízeních jako stolní počítač, tablet, mobil, konzole nebo televizi. Aplikace cílená na zařízení používající systém Android, který vychází ze systému Linux, jsou překládána pro verzi Android 8.0 nebo novější. Aplikace cílené na iOS v Xamarin.Forms jsou určené pro mobilní zařízení iPhone 5s a novější. Používající systém iOS 9.0 nebo novější. Pro vývoj aplikací pro iOS musí být Visual Studio spárované s počítačem, který je na stejné lokální síti a je na něm nainstalován desktopový systém macOS Sierra 10.12 nebo vyšší. Na tomto systému musí být nainstalován Visual Studio 2017 for Mac.

Xamarin.Forms je společný framework pro vývoj aplikací na všechny tři tyto platformy za použití jednoho sdíleného kódu a to i pro grafické rozhraní, které také využívá technologii datové vazby a XAML souborů popsaných výše. Tak zdrojový kód aplikace i aplikace samotná vychází

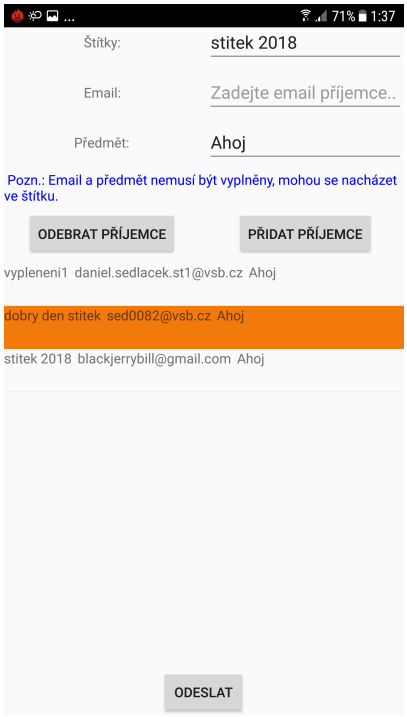
s WPF aplikace tlustého klienta.

Pro přístup k datům využívají mobilní klienti server, který je součástí prezentační vrstvy. Jde o konzolovou aplikaci, která přes logickou a datovou vrstvu skrze Entity Framework komunikuje s SQL serverem a získaná data poskytuje před TCP protokol klientům. Spojení inicializuje mobilní klient, který posílá mobilní požadavek na data a po navrácené odpovědi spojení končí. Požadavek i odpověď je ve formátu, který odpovídá standardu XML. Komunikace mezi klientem a serverem je šifrována symetrickou standardizovanou šifrou AES (Advanced Encryption Standard), jejíž původní název vzniklý ze jmen autorů je Rijndael. AES je rychlá a standardizovaná, je tak vhodná zároveň pro Android, iOS, UWP a konzolovou Windows aplikaci serveru. Mobilní klient je tenkou verzí WPF desktopové aplikace, který má všechny funkce jako tlustý klient, pouze bez možnosti náhledu na spojení šablony a jednotlivého štítků, tedy jednotlivých vytvořených emailů, šablony samotné k náhledu jsou.

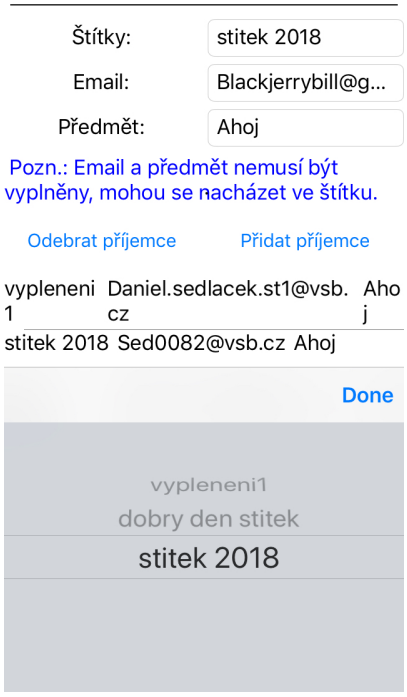
Obdobně jako u tlustého klienta se po spuštění mobilní aplikace zobrazí přihlašovací obrazovka. Po úspěšném přihlášení se načítá seznam šablon a štítků a první šablona se zobrazuje. Obrazovkou je možno rolovat. Pod zobrazenou šablonou může uživatel přidávat jednotlivé příjemce k odeslání se štítky. Takových příjemců může uživatel přidat neomezeně mnoho. Při odeslání příjemcům pošle mobilní klient požadavek na server, na něm probíhá sestavení jednotlivých emailů z jedné šablony a štítků a také následné odeslání sestavených zpráv. Na rozdíl od tlustého klienta, který odesílá zprávy přímo. Po odeslání je uživateli zobrazen počet úspěšně odeslaných emailů.



Obrázek 11: Ukázka mobilní aplikace pro platformu UWP při výběru šablony



Obrázek 12: Ukázka mobilní aplikace pro systém Android při editaci příjemců

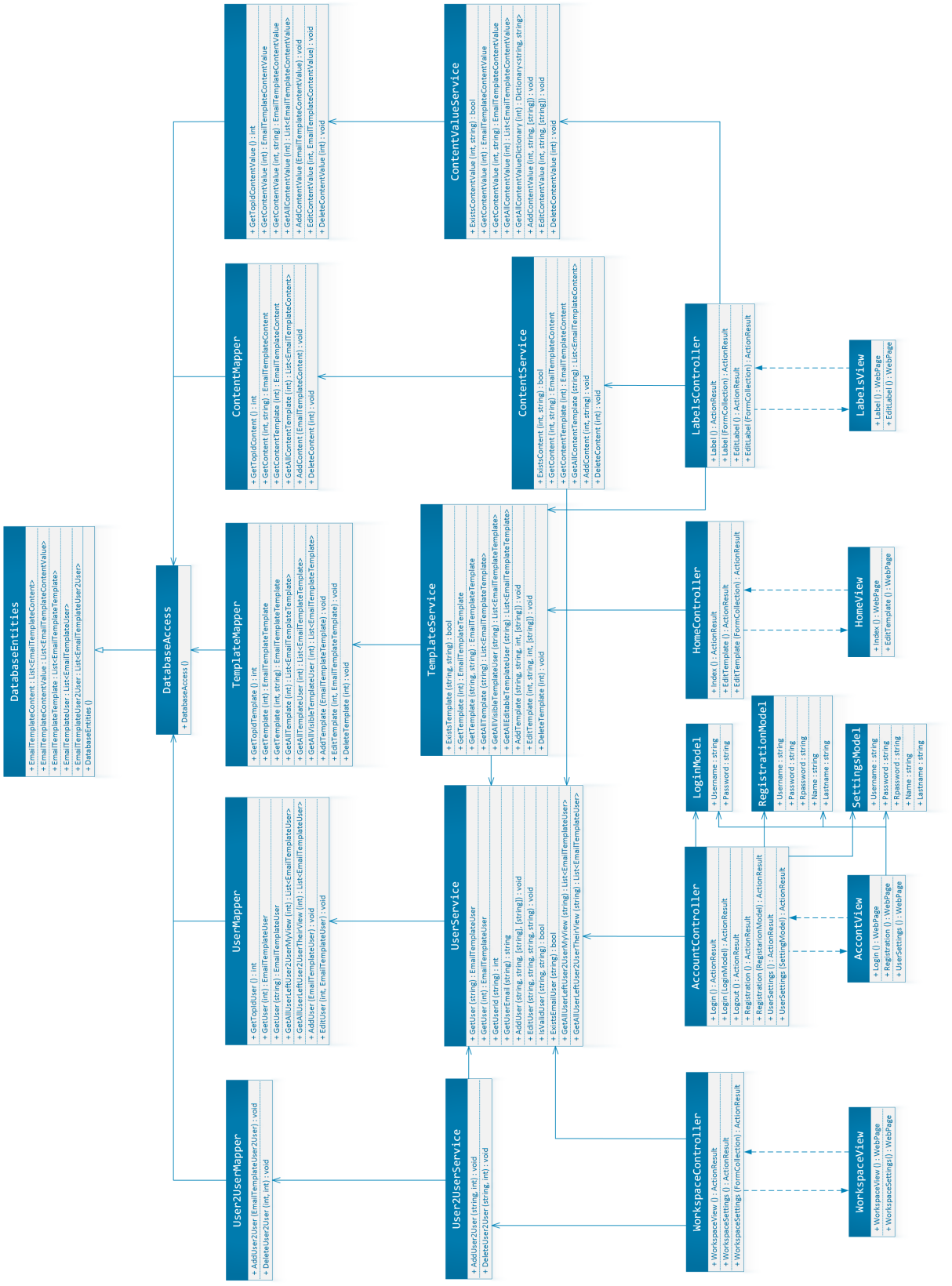


Obrázek 13: Ukázka mobilní aplikace pro systém iOS při výběru štítků

5.4 Implementace systému

Implementace systému je zobrazena diagramem implementace, který je diagramem tříd, viz. obrázek 14. Diagram zobrazuje statický pohled na systém, tj. zejména třídy jako typy objektů a statické vztahy, které mezi nimi existují. Na základě třídního diagramu by měl být programátor schopen naimplementovat strukturu systému. Třídní diagram obsahuje definici všech tříd aplikace a jejich atributů a metod. Diagram je platformově závislý, tedy specifický pro určitý programovací jazyk. Mimo jiné to znamená, že se v identifikátorech již nevyskytuje diakritika, atributy mají datové typy specifické pro daný jazyk a podobně.

Návrh systému mají na starosti zkušení programátoři a analytici. Díky návrhu se potom samotné programování může předat i méně zkušeným programátorům. Diagram tříd je dobré vytvořit však i v případě, že systém píše jen jedna osoba, donutí nás to nejprve přemýšlet v kontextu celého systému a až potom programovat. Nestane se, že se napíše velká část systému a potom se zjistí, že takhle nepůjde naimplementovat. Diagram zároveň slouží i jako dokumentace.

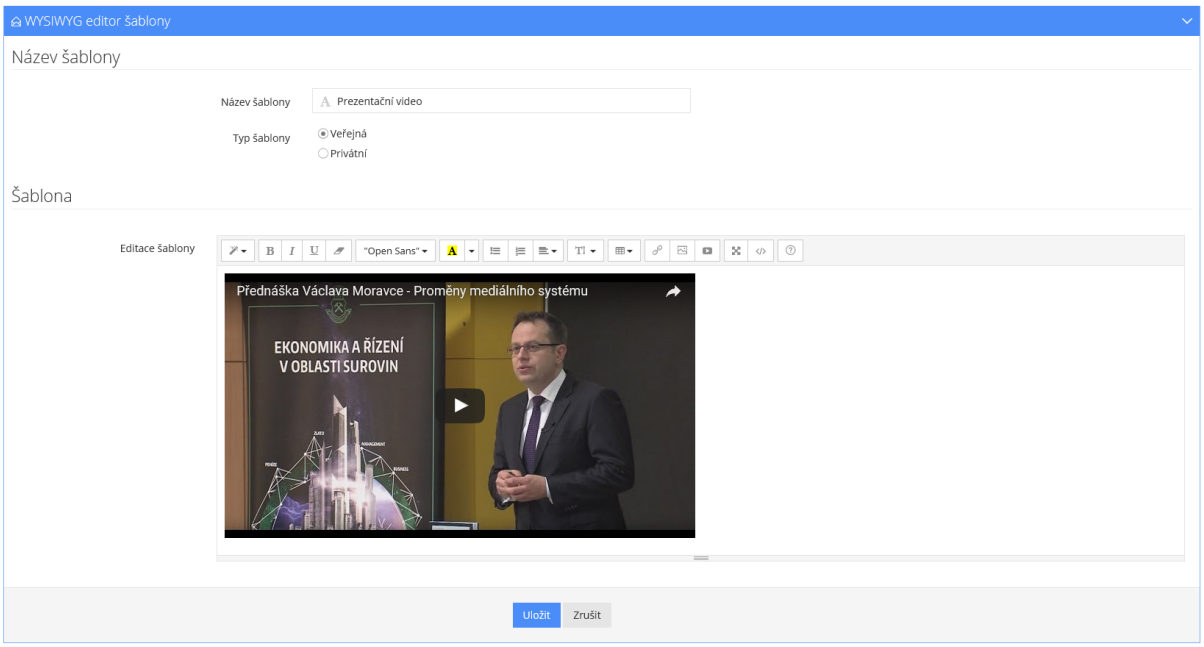


Obrázek 14: Třídní diagram webové části systému

5.5 Ukázka webové aplikace informačního systému

Po dokončení tvorby celého systému máme možnost porovnat jeho finální podobu s podobou plánovanou v prvních kapitolách. Ukážeme, jak aplikace reálně vypadá z pohledu uživatele ve webovém prohlížeči.

Na obrázku 15 je detail WYSIWYG editoru šablony. Ten podporuje změnu fontu písma a její velikosti. Standardní možnost typů písma tučné (Bold), kurzíva (Italic), podtržené (Underline). Podpora tvorby tabulek a nadpisů. Obecně změnu barvy prvku či jeho pozadí. V editoru se dají vytvářet nečíslované i číslované seznamy. Formátuje odstavce do 6-ti různých variant. Dají se přidávat obrázky a různá flash videa, konkrétně od poskytovatelů YouTube, Vimeo, Vine, Instagram nebo DailyMotion. Pro zkušené uživatele je možnost přepnout do přímého zobrazení HTML kódu šablony.



Obrázek 15: Náhled systému, editace šablony

6 Přechod

Cílem fáze přechod je nasazení systému do prostředí Internetu, což by měla být činnost s co nejmenším narušením běžné obchodní činnosti vykonávané organizací, která systém integruje do svých struktur. Jde o souhrn všech činností, po jejímž dosažení je software plně funkční a dostupný k použití. V následujících kapitolách se nachází konkrétní popis pro přípravu a nasazení systému.

6.1 Uvolnění a publikace aplikace

Nejdříve je nutné aplikaci uvolnit, tedy vydat její finální verzi. Finální verze vznikla již ve fázi rychlá výstavba, nicméně nebyla ještě uvolněna. Dosud probíhala výstavba informačního systému ve Visual Studiu v režimu ladění (Debug), tento režim je určený pro vývoj, určuje jakým způsobem bude aplikace přeložena a sestavena. Takto přeložená aplikace je vhodná pro testování a vývoj, protože umožňuje při výskytu chyby i při ladění systému přerušovat aplikaci po jednotlivých krocích, instrukcích. A zjišťovat při tom hodnoty proměnných a tok programu, což umožní vývojáři najít konkrétní místo v kódu. Tento režim je tak velmi silným nástrojem vývojového prostředí, samozřejmě za odpovídající ztrátu výkonu aplikace. Pro uvolnění aplikace přepneme kompilaci do režimu Release (Vydání), v tomto režimu bude celý systém sestaven. Jeho různé části tak budou připraveny k nasazení. Jde o verzi, která kromě svého samotného obsahu neobsahuje žádný kód navíc oproti režimu ladění. Jde o tedy nejrychlejší a nejmenší možnou verzi aplikace.

U WPF desktopové aplikace vznikne přímo spustitelný soubor bez nutnosti instalace, ten bude zveřejněn na webovém portále, kde ho můžou stáhnou i neregistrovaní uživatelé. Webová aplikace bude sestavena pro nasazení na webovém serveru. Poté je nutné webovou aplikaci informačního systému publikovat, to se provede také ve Visual Studiu. Jde o proces vytvoření odpovídající struktury celého systému pro webový server. Pro aplikaci byla vybrána metoda nasazení a publikace metodou souborového systému (File System), publikace proběhla pro webový server Internetové informační služby (Internet Information Services). Vytvořila se tak složka, jejíž nově vzniklý obsah je publikovaná uvolněná aplikace webového serveru, vhodná pro provoz na IIS.

6.2 Instalace serveru a nasazení

Pro provoz a hostování webové části aplikace byl vybrán operační systém společnosti Microsoft Windows 2016 Server Datacenter, jedná se o serverový operační systém navržený pro provoz webových aplikací, schopný obsloužit velké množství požadavků, je tak vhodný pro tuto aplikaci při nasazení ve středních a velkých organizacích. Tento systém obsahuje IIS 10, nicméně není nainstalován. Po instalaci operačního systému se spustí správce serveru (Server Manager) a dá se přidat role nebo funkce (Add roles and features), zde se vybere IIS (Web Server). Dále

se upraví funkce (features) IIS, webový server musí podporovat ASP.NET 4.6 a skupinovou správu zásad (Group policy management). Následně se nastaví role (Role services), zahrnují se všechny role běžných funkcí protokolu HTTP (Common HTTP features), dále všechny funkce bezpečnosti (Security) a možnost filtrování požadavků (Request Filtering) a základního ověření (Basic Authentication). Doporučuji nainstalovat celý balíček nástrojů pro správu (Management Tools) pro možnost správy serveru v grafickém uživatelském prostředí.

Po instalaci IIS se přidá webová aplikace vytvořená dříve. Ve správci IIS se přidá aplikace (Add Application), napíšeme její webové jméno a specifikujeme adresář cesty, kde bude aplikace umístěna. Do tohoto adresáře se zkopíruje obsah složky s publikovanou uvolněnou aplikací a tím se aplikace nasadí na webový server.

6.3 Databázový server

Pro nasazení je nutné zprovoznit relačně databázový server, který byl vybrán ve fázi vývoje. Microsoft SQL Server 2016 Standard budeme hostit na stejném operačním systému jako webový server, nicméně na jiném hardwaru z důvodu rozložení zátěže, lepší dostupnosti a schopnosti obnovy při výpadku. Po instalaci SQL serveru stačí vytvořit příslušného uživatele s příslušnými právy a databázi pro internetovou aplikaci. Struktura databáze se vytvoří jednoduše spuštěním skriptu v jazyku pro definici dat (Data definition language), který je součástí přílohy softwarové části.

6.4 Aktualizace

Způsob aktualizace byl předdefinován volbou publikace aplikace metodou souborového systému. Při změně ve vývojové části dojde po otestování změn k opětovné publikaci uvolněné aplikace webového informačního systému. Ta nahradí obsah specifického adresáře s původní aplikací na serveru. Při změně databáze se musí vytvořit nová databáze podle nového skriptu pro její tvorbu. A musí tak vzniknou netriviální množina funkcí a SQL příkazů, která původní data převádí do nové podoby struktury databáze.

7 Závěr

Cílem této bakalářské práce bylo vytvořit informační systém pro vytváření a správu šablon elektronických zpráv, jehož součástí je webový portál pro tvorbu a editaci šablon a celkovou správu uživatele. Dále desktopová aplikace tlustého klienta pro Windows a mobilní aplikace pro Android, iOS a UWP tenkého klienta, které slouží k odesílání elektronických zpráv sestavených na základě šablony.

V první části práce jsem určoval požadavky na systém jak funkční tak nefunkční. Poté jsem se věnoval návrhu systému. Při rešerši problému jsem prostudoval možné varianty systémové architektury, která určuje celkovou strukturu systému i strukturu dílčích prvků. Myslím, že třívrstvá architektura byla vhodně zvolena vzhledem k rozsahu bakalářské práce s možností dalšího rozšíření. Zpětný náhled na analýzu požadavků při plánování systému, které jsou vyjádřené třeba skicou budoucího systému, při porovnání s hotovým systémem se ukazuje, že finální verze systému je velmi podobná původnímu návrhu. Rozdíl mezi návrhem a finální podobou je způsoben změnou designu systému při jeho iteračním vývoji za kontinuálních konzultací s uživateli. Taková změna návrhu a implementace je v rámci životního cyklu aplikace popsaného v metodice vývoje Rapid application development očekávaná.

Při tvorbě webového portálu za použití MVC jsem se naučil používat dnes velmi rozšířenou technologii. K vývoji desktopové aplikace byla zvolena moderní platforma WPF pro grafická uživatelská prostředí. Pro tvorbu mobilních aplikací byla zvolena multiplatformní technologie Xamarin.Forms. Při implementaci datové vrstvy jsem se seznámil s Entity Framework. Zvolil jsem odpovídající návrhové vzory. Nakonec jsem naimplementoval systém a popsal všechny klíčové technologie.

Webová aplikace informačního systému obsahuje nástroje na tvorbu šablon elektronických zpráv. Nástroje umožňují pohodlnou, uživatelsky přívětivou editaci šablon s prvky jako tabulka, číslovaný a nečíslovaný seznam a např. s možností přidávat flash videa různých vydavatelů jako YouTube, Instagram, atd.

Další nezbytnou součástí systému je sada knihoven umožňující vytvoření elektronických zpráv z předem uložené šablony na webovém portále a její následné odeslání příjemcům. Tyto knihovny jsou použity pro tvorbu desktopové i mobilní aplikace. Knihovny i desktopová aplikace jsou dostupné ke stažení na portále.

Vytvořený systém vyniká na trhu tím, že umožňuje neomezenou tvorbu šablon s neomezeným počtem příjemců. S možností zdarma stáhnout šablony ostatních uživatelů a s možností rozšíření do dalších aplikací. Systém umožňuje zdarma používat desktopovou aplikaci i mobilní klienty na všechny běžné mobilní platformy.

Při vypracovávání této bakalářské práce bylo rozhodující se naučit pracovat se všemi dílčími technologiemi a svázat je dohromady. Byly splněny všechny cíle této práce, tvorba softwarové analýzy, návrhu a následná tvorba samotného softwarového díla. Funkčnost aplikace byla ověřena jejím nasazením.

Literatura

[1] TROELSEN, Andrew. *Professional C# 5.0 and the .NET 4.5 Framework 6th Edition*. Berkely, CA, USA: Apress, 2012. ISBN 1430242337.

[2] BISHOP, Judith. *C# 3.0 Design Patterns*. Sebastopol, CA: O'Reilly Media, 2008. ISBN 978-0-596-52773-0.

[3] MARTIN, James. *Rapid application development*. 1. Indianapolis, IN, USA: Macmillan Publishing Co., 1991. ISBN 0-02-376775-8.

[4] FOWLER, Martin. *Patterns of Enterprise Application Architecture*. 7th. US: Addison-Wesley Professional, 2012. ISBN 0-321-12742-0.

[5] VERSLUIS, Gerald. *Xamarin.Forms Essentials: First Steps Toward Cross-Platform Mobile Apps*. CA, USA: Apress, 2017. ISBN 978-1-4842-3239-2.

[6] SNIDER, Ed. *Mastering Xamarin.Forms - Second Edition*. 2nd Revised edition. Birmingham, UK: Packt Publishing, 2018. ISBN 978-1-78829-026-5.

[7] YUEN, Sheridan. *Mastering Windows Presentation Foundation*. Birmingham, UK: Packt Publishing, 2017. ISBN 978-1-78588-300-2.

[8] RICHARDS, Mark. *Software Architecture Patterns*. Sebastopol, CA, USA: O'Reilly Media, 2015. ISBN 978-1-491-92424-2.

[9] JABLONSKI, Stefan, Ilia PETROV, Christian MEILER a Udo MAYER. *Guide to Web Application and Platform Architectures*. Berlin Heidelberg New York: Springer, 2013. ISBN 978-3-642-05668-0.

[10] ESPOSITO, Antonio. *Learning .NET High-perfomance Programimng*. Birmingham, UK: Packt Publishing, 2015. ISBN 978-1-78528-846-3.

[11] FREEMAN, Adam. *Pro ASP.NET MVC 5*. Fifth Edition. CA, USA: Apress, 2013. ISBN 978-1-4302-6529-0.

[12] T. S. RAGUPATHI, Mugilan. *Learning ASP.NET Core MVC Programming*. Birmingham, UK: Packt Publishing, 2016. ISBN 978-1-78646-383-8.

[13] NAYLOR, Lee. *ASP.NET MVC with Entity Framework and CSS*. CA, USA: Apress, 2016. ISBN 978-1-4842-2136-5.

[14] VAN DER WESTHUIZEN, Pieter. *Bootstrap for ASP.NET MVC*. Second Edition. Birmingham, UK: Packt Publishing, 2016. ISBN 978-1-78588-947-9.

[15] Vícevrstvá architektura: popis vrstev, http://www.cleverandsmart.cz/vicevrstva-architektura-popis-vrstev/, poslední úprava 9.5.2013

[16] Zářná budoucnost Entity Framework Code First, https://www.miroslavholec.cz/blog/entity-framework-code-first-zarna-budoucnost, poslední úprava 29.3.2016

[17] Vícevrstvé architektury aplikací, http://www.robertdresler.cz/2011/04/vicevrstve-architektury-aplikaci.html, poslední úprava 24.4.2011

[18] Jak uplatnit principy třívrstvé architektury v rámci web integračního projektu, http://www.web-integrator.cz/webova-integrace/jak-uplatnit-principy-trivrstve-architektury-v-ramci-web-integracniho-projektu/52-jak-uplatnit-principy-trivrstve-architektury-v-ramci-web-integracniho-projektu, poslední úprava 15.11.2013

[19] MVC architektura, http://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury/, poslední úprava 2014

[20] Historie a vývoj jazyka C (od C až po C#), http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xcernoh1.h poslední úprava 2003

[21] ASP.NET?, http://www.aspnet.cz/articles/5417-quo-vadis-asp-net, poslední úprava 3.1.2014